ATSC Standard:
Content Recovery in Redistribution Scenarios

Doc. A/336:2022-03
31 March 2022

Advanced Television Systems Committee
1300 I Street, N.W., Suite 400E
Washington, D.C. 20005
202-872-9160

The Advanced Television Systems Committee, Inc. is an international, non-profit organization developing voluntary standards and recommended practices for broadcast television and multimedia data distribution. ATSC member organizations represent the broadcast, professional equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries. ATSC also develops implementation strategies and supports educational activities on ATSC standards. ATSC was formed in 1983 by the member organizations of the Joint Committee on Inter-society Coordination (JCIC): the Consumer Technology Association (CTA), the Institute of Electrical and Electronics Engineers (IEEE), the National Association of Broadcasters (NAB), the Internet & Television Association (NCTA), and the Society of Motion Picture and Television Engineers (SMPTE). For more information visit www.atsc.org.

Note: The user's attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights. By publication of this standard, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. One or more patent holders have, however, filed a statement regarding the terms on which such patent holder(s) may be willing to grant a license under these rights to individuals or entities desiring to obtain such a license. Details may be obtained from the ATSC Secretary and the patent holder.

Implementers with feedback, comments, or potential bug reports relating to this document may contact ATSC at https://www.atsc.org/feedback/.

Revision History

Version	Date
Candidate Standard approved	15 January 2016
A/336:2017 Standard approved	24 February 2017
A/336:2017 Revision approved	5 June 2017
A/336:2018 Standard approved	23 April 2018.
A/336:2018 Revision approved	11 December 2018
Amendment No. 1 approved	26 August 2019
Amendment No. 2 approved	27 August 2019
Amendment No. 3 approved	3 October 2019
Text references with broken links fixed as needed	4 October 2019
A/336:2019 published	4 October 2019
Amendment No. 1 approved	4 June 2020
Corrigendum No. 1 approved	7 November 2021
A/336:2022-03 published (references to ATSC documents updated)	31 March 2022

Table of Contents

1. SCOPE 1
   1.1 Introduction and Background 1
   1.2 Organization 1
2. REFERENCES 2
   2.1 Normative References 2
   2.2 Informative References 3
3. DEFINITION OF TERMS 3
   3.1 Compliance Notation 3
   3.2 Treatment of Syntactic Elements 3
      3.2.1 Reserved Elements 3
   3.3 Acronyms and Abbreviations 4
   3.4 Terms 4
   3.5 Extensibility 5
4. SYSTEM OVERVIEW 6
   4.1 Watermarking 6
   4.2 Fingerprinting 8
5. SPECIFICATION 9
   5.1 Video Watermark Payload Format 9
      5.1.1 Watermark Message Syntax 10
      5.1.2 Multiplexing and Processing Rules 12
      5.1.3 Segmentation and Reassembly Examples 13
      5.1.4 Content ID Message 15
      5.1.5 Presentation Time Message 17
      5.1.6 URI Message 18
      5.1.7 VP1 Message 20
      5.1.8 Dynamic Event Message 23
      5.1.9 Display Override Message 25
      5.1.10 Advanced Emergency Alert Message 25
      5.1.11 User Private Message 29
   5.2 Audio Watermark Payload Format 30
      5.2.1 Packet 30
      5.2.2 Scrambling 30
      5.2.3 VP1 Payload 31
      5.2.4 Display Override State 32
      5.2.5 VP1 Audio Watermark Segment 33
      5.2.6 Examples (Informative) 33
   5.3 Fingerprint Methodology 33
      5.3.1 Introduction and Architecture 33
      5.3.2 Request/Response Model 33
      5.3.3 Event-Driven Model 34
   5.4 Recovery Process 34
      5.4.1 Recovery File Retrieval via Broadband 35
      5.4.2 Hostname Determination 36
      5.4.3 Recovery File Format 36
      5.4.4 Dynamic Event Retrieval via Broadband 41

5.4.5 AEAT Retrieval via Broadband 42
ANNEX A : IMPLICATIONS OF SET-TOP BOX OPERATIONS 43
ANNEX B : JSON RECOVERY FILE SCHEMA....45
ANNEX C : ACQUISITION OF SIGNALING USING VIDEO WATERMARKS....53
ANNEX D : ACQUISITION OF SIGNALING USING VP1 PAYLOADS/MESSAGES 54
ANNEX E : MEDIA TYPE REGISTRATIONS....56
E.1 RDT 56
E.2 DYNAMIC EVENT MESSAGE 57

Index of Figures and Tables

Figure 4.1 Architecture for direct watermark signaling approach. 7
Figure 4.2 Architecture for indirect watermark signaling approach. 8
Figure 4.3 Architecture for fingerprinting approach. 9
Figure 5.1 Illustration of message segmentation and interleaving (1X System). 14
Figure 5.2 Temporal structure of VP1 Message Groups carrying vp1_message(). 21
Figure 5.3 Temporal structure of VP1 Message Groups carrying extended_vp1_message(). 22
Figure 5.4 Temporal structure of VP1 Message Groups carrying both vp1_message() and extended_vp1_message(). 23

Table 5.1 Bit Stream Syntax of the Watermark Payload 9
Table 5.2 Bit Stream Syntax for the Watermark Message Block 10
Table 5.3 wm_message_id Encoding 11
Table 5.4 Bit Stream Syntax of the Reassembled Watermark Message 12
Table 5.5 Bit Stream Syntax for the Content ID Message 15
Table 5.6 content_ID_type Field Encoding 16
Table 5.7 Bit Stream Syntax for the Presentation Time Message 18
Table 5.8 Bit Stream Syntax for the URI Message 18
Table 5.9 uri_type field Encoding 19
Table 5.10 domain_code Encoding 19
Table 5.11 Bit Stream Syntax for the vp1_message() 20
Table 5.12 Bit Stream Syntax for the extended_vp1_message() 21
Table 5.13 Alternate Whitening Sequences 21
Table 5.14 Bit Stream Syntax for the Dynamic Event Message 23
Table 5.15 delivery_protocol_type field Encoding 24
Table 5.16 Bit Stream Syntax for the Display Override Message 25
Table 5.17 Bit Stream Syntax for the Advanced Emergency Alert Message 26
Table 5.18 Code Values for AEA_type Field 27
Table 5.19 Code Values for audience Field 27
Table 5.20 domain_code Encoding 28
Table 5.21 Bit Stream Syntax for the User Private Message 29
Table 5.22 Syntax of packet() Structure 30
Table 5.23 Whitening Sequences 31
Table 5.24 Polynomials used for BCH code (127, 50, 13) 31
Table 5.25 Syntax of payload() Structure 31
Table 5.26 Syntax small_domain() Structure 32
Table 5.27 Syntax large_domain() Structure 32
Table 5.28 Supported Range of Server Codes and Interval Codes per Domain Type 32
Table 5.29 Example cell() Sequences, Shown as Hexadecimal Values. 33
Table 5.30 Recovery File Format Logical Structure 36
Table 5.31 Component Description Type 40
Table 5.32 Component Anchor 41
Table A.1 Components Modified and Associated WM/FP Identification Capabilities for STB Functions 43

Table A.2 WM or FP Required to Identify STB Modifications for Receiver Service Functions 44

ATSC Standard: Content Recovery in Redistribution Scenarios

1. SCOPE

This document specifies methods by which receivers may recover content that is available from a broadcaster, but not delivered directly to the receiver through the RF broadcast signal. This method may be useful in the case when the receiver is connected to a set-top box via an HDMI cable and is only receiving uncompressed A/V content and not the full signaling tables or alternate content choices offered by the broadcaster.

The methods described in this document require an automatic content recognition (ACR) mechanism. The ACR mechanism is either the video and audio watermarks specified for use within ATSC 3.0 broadcast emissions ([5] and [4], respectively) or a fingerprint mechanism described in this document.

This document describes the method for a receiver to recover content via a broadband interface once it has acquired an ATSC 3.0 watermark payload or has identified content using fingerprints. It also specifies the format of certain files that may be returned in response to such a content recovery request.

1.1 Introduction and Background

A Redistribution scenario is a situation where an ATSC 3.0 receiver is getting TV programming through an intermediary such as a Multi-channel Video Program Distributor (MVPD) – e.g., a cable, satellite or IPTV operator. If there is supplementary content such as applications or alternate audio tracks that is not passed through by the intermediary, the original broadcaster of the content might want to make such supplementary content available via alternate means.

The document specifies how certain signaling information can be carried in audio and/or video watermark payloads, how content recognition can be done using fingerprints, and how this information can be used to access supplementary content in a redistribution scenario.

1.2 Organization

This document is organized as follows:

- Section 1 – Outlines the scope of this document.
- Section 2 – Lists normative and informative references.
- Section 3 – Defines terms, acronyms, and abbreviations for this document.
- Section 4 – Provides a system overview
- Section 5 – Specifies audio/video watermark payload and fingerprint methodology
- Annex A – Discusses implications of set-top box operations
- Annex B – Specifies the Recovery File JSON schema
- Annex C – Provides an informative discussion of acquisition of signaling using video watermarks
- Annex D – Provides an informative discussion of acquisition of signaling using VP1 payloads/messages
- Annex E – Media Type Registrations

2. REFERENCES

All referenced documents are subject to revision. Users of this Standard are cautioned that newer editions might or might not be compatible.

2.1 Normative References

The following documents, in whole or in part, as referenced in this document, contain specific provisions that are to be followed strictly in order to implement a provision of this Standard.

[1] ATSC: "ATSC Standard: Signaling, Delivery, Synchronization and Error Correction," Doc. A/331:2022-03, Advanced Television Systems Committee, Washington, DC, 31 March 2022.

[2] ATSC: "ATSC Standard: Service Announcement," Doc. A/332:2022-03, Advanced Television Systems Committee, Washington, DC, 31 March 2022.

[3] ATSC: "ATSC Standard: Service Usage Reporting," Doc. A/333:2022-03, Advanced Television Systems Committee, Washington, DC, 31 March 2022.

[4] ATSC: "ATSC Standard: Audio Watermark Emission," Doc. A/334:2022-03, Advanced Television Systems Committee, Washington, DC, 31 March 2022.

[5] ATSC: "ATSC Standard: Video Watermark Emission," Doc. A/335:2022-03, Advanced Television Systems Committee, Washington, DC, 31 March 2022.

[6] ATSC: "ATSC Standard: Application Signaling," Doc. A/337:2022-03, Advanced Television Systems Committee, Washington, DC, 31 March 2022.

[7] ATSC: "ATSC Standard: ATSC 3.0 Security and Service Protection," Doc. A/360:2022-03, Advanced Television Systems Committee, Washington, DC, 31 March 2022.

[8] IEEE: "Use of the International Systems of Units (SI): The Modern Metric System," Doc. SI 10, Institute of Electrical and Electronics Engineers, New York, NY.

[9] IETF: "Domain Names – Implementation and Specification," RFC 1035, Internet Engineering Task Force, November 1987.

[10] IETF: "Uniform Resource Identifiers (URI): Generic Syntax," RFC 3986, Internet Engineering Task Force, January 2005.

[11] IETF: "The 'tag' URI Scheme", RFC 4151, Internet Engineering Task Force, October 2005.

[12] ISO: "Codes for the representation of names of countries and their subdivisions — Part 1: Country codes," Doc. ISO 3166-1:2013 (E/F), International Organization for Standardization.

[13] ISO/IEC: "Information technology – Generic coding of moving pictures and associated audio – Part 1: Systems," Doc. ISO/IEC 13818-1:2015, International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[14] ISO/IEC: "Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats," Doc. ISO/IEC 23009-1:2014, International Organization for Standardization/ International Electrotechnical Commission, Geneva Switzerland.

[15] SMPTE: "Digital Object Identifier (DOI) Name and Entertainment ID Registry (EIDR) Identifier Representations," RP 2079-2013, Society of Motion Picture and Television Engineers, White Plains, NY, 2013.

[16] SMPTE: "Advertising Digital Identifier (Ad-ID®) Representations," RP 2092-1, Society of Motion Picture and Television Engineers, White Plains, NY, 2015.

[17] IETF: "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 8259, Internet Engineering Task Force, December 2017. https://tools.ietf.org/html/rfc8259

2.2 Informative References

The following documents contain information that may be helpful in applying this Standard.

[18] ATSC: "ATSC Standard: Interactive Content," Doc. A/344:2022-03, Advanced Television Systems Committee, Washington, D.C., 31 March 2022.

[19] ATSC: "ATSC Standard: Interactive Services Standard," Document A/105:2015, Advanced Television Systems Committee, Washington, DC, 29 October 2015.

[20] IETF: "Media Type Specifications and Registration Procedures," RFC 6838 (BCP 13), Internet Engineering Task Force, January 2013. https://tools.ietf.org/html/rfc6838

[21] ATSC: "Code Point Registry", Advanced Television Systems Committee, Washington, D.C. https://www.atsc.org/standards/code-point-registry/

3. DEFINITION OF TERMS

With respect to definition of terms, abbreviations, and units, the practice of the Institute of Electrical and Electronics Engineers (IEEE) as outlined in the Institute's published standards [8] shall be used. Where an abbreviation is not covered by IEEE practice or industry practice differs from IEEE practice, the abbreviation in question will be described in Section 3.3 of this document.

3.1 Compliance Notation

This section defines compliance terms for use by this document:

shall – This word indicates specific provisions that are to be followed strictly (no deviation is permitted).

shall not – This phrase indicates specific provisions that are absolutely prohibited.

should – This word indicates that a certain course of action is preferred but not necessarily required.

should not – This phrase means a certain possibility or course of action is undesirable but not prohibited.

3.2 Treatment of Syntactic Elements

This document contains symbolic references to syntactic elements used in the audio, video, and transport coding subsystems. These references are typographically distinguished by the use of a different font (e.g., restricted), may contain the underscore character (e.g., sequence_end_code) and may consist of character strings that are not English words (e.g., dynrng).

3.2.1 Reserved Elements

One or more reserved bits, symbols, fields, or ranges of values (i.e., elements) may be present in this document. These are used primarily to enable adding new values to a syntactical structure without altering its syntax or causing a problem with backwards compatibility, but they also can be used for other reasons.

The ATSC default value for reserved bits is '1.' There is no default value for other reserved elements. Use of reserved elements except as defined in ATSC Standards or by an industry standards setting body is not permitted. See individual element semantics for mandatory settings

and any additional use constraints. As currently-reserved elements may be assigned values and meanings in future versions of this Standard, receiving devices built to this version are expected to ignore all values appearing in currently-reserved elements to avoid possible future failure to function as intended.

ATSC Reserved indicates that only ATSC has currently defined values. Industry Reserved indicates that other SDOs have defined values and users are advised to consult the ATSC Code Point Registry [21].

3.3 Acronyms and Abbreviations

The following acronyms and abbreviations are used within this document.

Ad-ID – Format identifier for identifying advertising assets

ATSC – Advanced Television Systems Committee

BSID – Broadcast Stream ID

BCH(n, k, t) – Bose-Chaudhuri-Hocquenghem Error Correction Code having an n-bit codeword with k information bits (and hence n-k parity check bits) that is able to correct up to t bit-errors in the information bits.

bslbf – Bit string, left bit first.

EIDR – Entertainment Identifier Registry

HDMI – High-Definition Multimedia Interface

LSB – Least significant bit.

MSB – Most significant bit.

MVPD – Multi-channel Video Program Distributor

SMPTE – Society of Motion Picture and Television Engineers

uimsbf – Unsigned integer, most significant bit first.

URI – Uniform Resource Identifier

3.4 Terms

The following terms are used within this document.

Broadcast Stream ID – Identifier of a broadcast stream, as defined in Section 6.3.2 of A/331 [1].

Broadcast Stream – The abstraction for an RF Channel which is defined in terms of a carrier frequency centered within a specified bandwidth.

Cell – Has the meaning given in the ATSC A/334 Audio Watermark Emission specification [4].

Dynamic Event – Has the meaning given in A/337 [6].

I(x) mod G(x) – The remainder that results from the polynomial division of I(x) by G(x).

Interval Code – A value that identifies the interval of content in which the VP1 payload value is embedded.

Large Domain – A division of the VP1 Payload wherein the interval field has a size of 25 bits, sufficient to support watermarking of approximately 1 year, 7 months of content.

Major Channel Number – Part of the identification of a service, as defined in Section 6.3.2 of A/331 [1] as `SLT.Service@majorChannelNo`. The combination of major and minor channel number is unique within a broadcast area. I.e., a receiver of over-the-air broadcasts will never see two different services with the same major/minor channel number combination.

Minor Channel Number – Part of the identification of a service, as defined in Section 6.3.2 of A/331 [1] as `SLT.Service@minorChannelNo`.

reserved – Set aside for future use by a Standard.

Packet – Has the meaning given in the ATSC A/334 Audio Watermark Emission specification [4].

Parity – A division of the Cell that carries the BCH code parity check bits.

Query Flag – The value of the query_flag field in an instance of the VP1 Payload.

Recovery File – A file containing signaling data about the service being presented including URLs that can be used to access signaling information for supplementary signaling and content. See also Section 5.4.3.

Recovery File Server – A server that provides Recovery Files to receivers.

Recovery Media Timeline – A linear frame of reference established using automatic content recognition (ACR) that associates positional time values to content in a media presentation. For ROUTE/DASH services, this corresponds to the DASH media presentation timeline. For MMT services, this corresponds to the MMT presentation timeline.

Server Code – A value that identifies a server which acts as the starting point for acquisition of supplementary content. (See Section 5.2.3.)

Signaling Server – A server that provides access to the Service Layer Signaling (SLS), as specified in Section 7 of A/331 [1].

Small Domain – A division of the VP1 Payload wherein the interval field has a size of 17 bits, sufficient to support watermarking of approximately 54 hours of content.

VP1 – The audio watermarking technology standardized in the ATSC A/334 Audio Watermark Emission specification [4].

VP1 Audio Watermark Segment – A continuous interval of watermarked audio during which signaling information recovered via audio watermark Payloads is applicable.

VP1 Message – A vp1_message() or an extended_vp1_message().

VP1 Message Group – A set of video frames, each carrying a VP1 Message, each carrying a packet() with an identical VP1 Payload.

VP1 Payload – The specific arrangement of the 50-bit VP1 payload in domain_type, server_field, interval_field, and query_flag specified in the present document.

VP1 Video Watermark Segment – A continuous interval of time that contains watermarked video during which signaling information recovered via VP1 Message is applicable.

VP1 Watermark Segment – A VP1 Video Watermark Segment or a VP1 Audio Watermark Segment.

3.5 Extensibility

The ATSC watermark system is designed such that new message types and new data elements can be added at a later time without adverse impact to existing receivers. For the video watermark system:

- The syntax of the wm_message_block() includes a wm_message_id field. While the current standard defines a number of message types, future revisions of the standard could define additional types. Existing receivers are expected to use the wm_message_block_length field to gracefully discard any message types they do not recognize or support.
- The wm_message_block() may be extended in the future with new fields at the end (just in front of CRC_32). Legacy receivers would be expected to disregard the added fields.

- A video watermark with a different run-in pattern could be developed in the future. Legacy receivers would not recognize the new mark and treat it as unmarked content.
- In the content_id_message(), two different identifier schemes are specified in the present standard, EIDR and Ad-ID. In the future the content_id_message() could carry identifiers of other types. Receivers not recognizing or supporting the new types are expected to discard the content_id_message().

For the audio watermark system

- An alternate Packet format may be defined to carry a different payload in redistribution content. Note that an alternate Packet format may not be decodable by existing receivers and it may be required to define a different or expanded emission specification in order for the alternate Packet format to be capable of being embedded simultaneously in content with the existing payload.
- New network protocols may be defined for receivers to obtain recovery and signaling information from Signaling Servers. Such protocols may be used in new receivers as replacements for the existing protocols or for use in addition to the existing protocols to provide new functionality.
- New data fields may be defined for the existing recovery and signaling files. Such data fields may be employed by new receivers to provide additional functionality.
- New data values may be defined for elements in the existing recovery and signaling files. Such data values may be employed, for example, to enable the identification of new types of signaling files within the Recovery File which may be retrieved by new receivers.

4. SYSTEM OVERVIEW

Signaling and supplemental content delivered over-the-air may not reach an ATSC 3.0 device receiving audio and video from a set-top box via an HDMI cable. This document describes methods for recovering ATSC 3.0 supplemental content in this or similar scenarios.

Content recovery can be achieved by delivering data to the ATSC 3.0 device via the HDMI connection, such as within an uncompressed audio or video signal. Such data minimally includes a means of identifying the content and may include additional information depending on the capacity of the data delivery method. The ATSC 3.0 receiver may then use this data to connect to a remote server via broadband that interprets the data and delivers the supplemental content to the receiver via broadband.

4.1 Watermarking

One such method of delivering data is via watermarks that can be extracted from uncompressed audio or video in the receiver. The methods of inserting watermarks for ATSC 3.0 content recovery are normatively specified in ATSC A/335 Video Watermark Emission specification [5] and ATSC A/334 Audio Watermark Emission specification [4].

There are two distinct processes for supplemental content recovery using watermarking discussed in this Standard.

One of these processes, “direct watermark signaling,” can be used when the content contains a full set of ATSC video watermarks. These watermarks contain instructions to access Signaling Servers, which provide data (such as MPDs and AEIs) needed to access and present the supplementary content. An informative description of example receiver behavior when this mode is employed is provided in Annex C of this Standard.

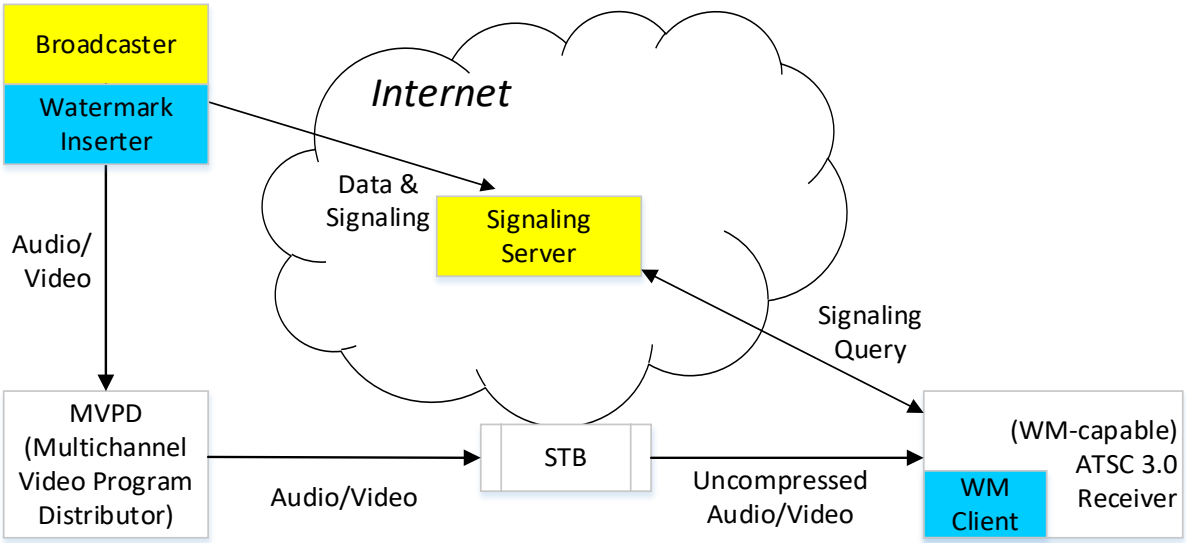

Figure 4.1 Architecture for direct watermark signaling approach.

The other process, "indirect watermark signaling," can be followed when the received content contains VP1 Payloads in video or audio watermarks. An informative description of example receiver behavior when this mode is employed is provided in Annex D of this Standard.

The primary difference between the direct and indirect watermark signaling processes is that the indirect process entails retrieving a Recovery File (as defined in Section 5.4.3) from a Recovery File Server, while the direct signaling process does not (because the full ATSC video watermark contains signaling data associated with the current service that enables the receiver to access the Signaling Server directly). To enable receivers to obtain IP addresses of Recovery File Servers associated with the Server Codes carried in VP1 Payloads, an operational VP1 DNS authoritative name server must exist.

In the case of indirect watermark signaling, the duration of time in the received content during which the recovered signaling information is applicable is established by VP1 Watermark Segments as described in Sections 5.1.7 (video) and 5.2.5 (audio), and illustrated in Figure 4.2. The presence of discontinuities in VP1 Watermark Segments can be used by the receiver, in addition to changes in the data contained in watermark payloads or messages, to identify service changes (e.g. tune in, channel change, component change).

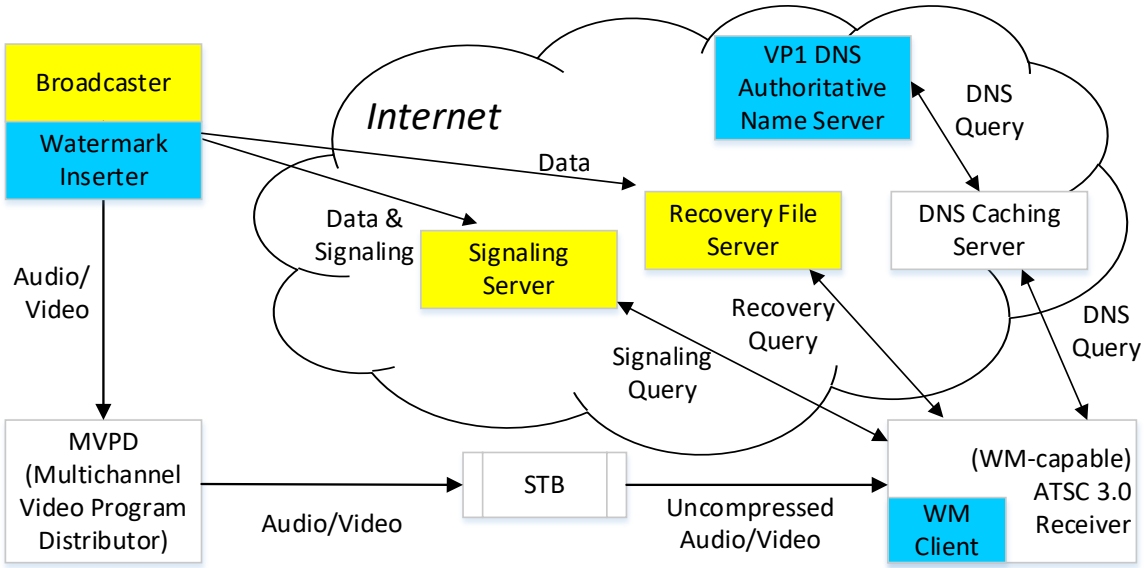

Figure 4.2 Architecture for indirect watermark signaling approach.

4.2 Fingerprinting

Another method of determining content identification is fingerprinting, which is introduced in the ATSC A/105 Interactive Services specification [19]. In this method, the receiver extracts fingerprints (also called signatures) from audio or video frames and sends these via broadband to a remote fingerprint server. The remote fingerprint server can check the fingerprints against a database of fingerprints from multiple broadcasters to identify the service where the frame occurred and when it occurred there. Extracting fingerprints and sending them to the remote fingerprint server can be done using proprietary technologies jointly agreed to by a fingerprint technology provider, a broadcaster and a receiver manufacturer, and thus this process is not normatively specified in the ATSC 3.0 standard. If the fingerprint is recognized, the fingerprint server returns a URL and media timing information so that the receiver is able to retrieve the supplementary content from a metadata server. A method for supplementary content retrieval is specified in Section 5.3.

Figure 4.3 below illustrates the system architecture of the fingerprinting approach to supplemental content recovery.

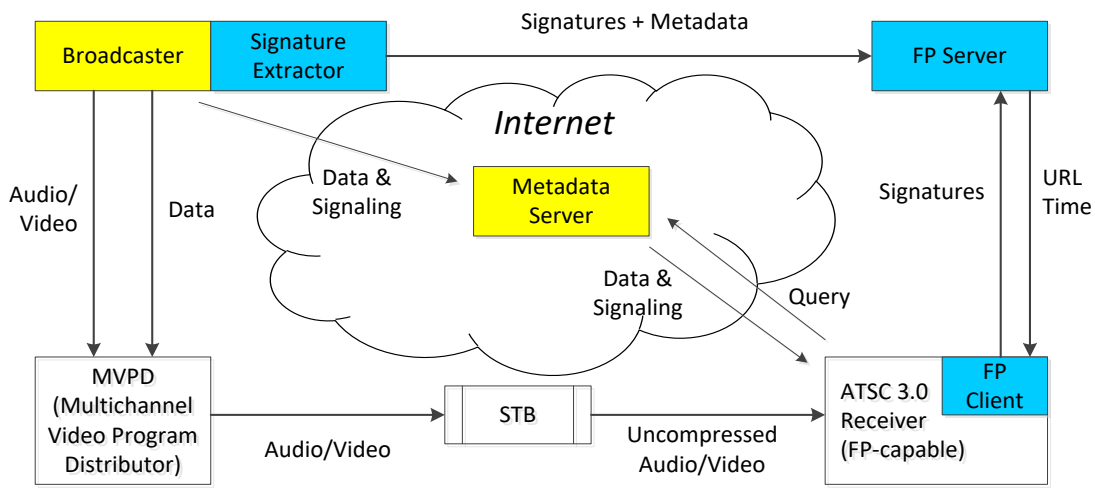

Figure 4.3 Architecture for fingerprinting approach.

5. SPECIFICATION

5.1 Video Watermark Payload Format

The emission format for video watermarks shall conform to the ATSC A/335 Video Watermark Emission specification [5]. As described in [5], the "1X" emission format delivers 30 bytes of data per video frame, while the "2X" system delivers 60 bytes per frame.

The VP1 Message (Section 5.1.7), when used, is repeated across multiple video frames. In instances where a watermark payload is not recovered from an individual video frame, a receiver may attempt to recover a VP1 Message by combining luma values (e.g. via averaging) from two or more successive video frames.

The payload format for the video watermark is the same in both the 1X and 2X systems. The run-in pattern (as specified in Section 5.1 of [5]) is followed by one or more instances of a message block. The watermark payload shall conform to the syntax given in Table 5.1.

Table 5.1 Bit Stream Syntax of the Watermark Payload

Syntax	No. of Bits	Format
watermark_payload() {		
run_in_pattern	16	0xEB52
for (i=0; i<N; i++) {		
wm_message_block()	var	
}		
for (i=0; i<M; i++) {		
zero_pad	8	0x00
}		
}		

run_in_pattern – This 16-bit value is set to 0xEB52 to identify that the video line includes a watermark of the format specified herein.

wm_message_block() – A full wm_message() or a fragment of a larger wm_message(), formatted per Section 5.1.1 below. The re-assembly of a wm_message() from multiple wm_message_block() instances shall be as described in Table 5.4.

zero_pad – A value of zero used to pad to the end of the frame.

5.1.1 Watermark Message Syntax

Watermark message blocks shall follow the syntax given in Table 5.2 below and the semantics that follow.

Table 5.2 Bit Stream Syntax for the Watermark Message Block

Syntax	No. of Bits	Format
wm_message_block() {		
wm_message_id	8	uimsbf
wm_message_block_length	8	uimsbf
wm_message_version	4	uimsbf
if ((wm_message_id & 0x80)==0) {		
fragment_number	2	uimsbf
last_fragment	2	uimsbf
} else {		
reserved	4	'1111'
fragment_number	8	uimsbf
last_fragment	8	uimsbf
}		
wm_message_bytes()	var	
if ((fragment_number == last_fragment) && (fragment_number != 0)) {		
message_CRC_32	32	uimsbf
}		
CRC_32	32	uimsbf
}		

wm_message_id – This 8-bit value shall uniquely identify the syntax and semantics of the data bytes carried in the message block, coded according to Table 5.3 below. As indicated in the table, the encodings of the watermark message types defined in the present standard may be found in Sections 5.1.4 through 5.1.11. Note that additional watermark message types might exist as indicated in the ATSC Code Point Registry [21].

Table 5.3 wm_message_id Encoding

wm_message_id Value	Max Fragments	Message	Reference
0x00	4	ATSC Reserved	
0x01		content_id_message()	Sec. 5.1.4
0x02		presentation_time_message()	Sec. 5.1.5
0x03		uri_message()	Sec. 5.1.6
0x04		vp1_ message()	Sec. 5.1.7
0x05		dynamic_event_message()	Sec. 5.1.8
0x06		display_override_message()	Sec. 5.1.9
0x07		extended_vp1_message()	Sec. 5.1.7
0x08-0x7E		Industry Reserved. See ATSC Code Point Registry [21].	See ATSC Code Point Registry [21]
0x7F		user_private_message()	Sec. 5.1.11
0x80	256	AEA_message()	Sec. 5.1.10
0x81		dynamic_event_message()	Sec. 5.1.8
0x82-0xFE		Industry Reserved. See ATSC Code Point Registry [21]	See ATSC Code Point Registry [21]
0xFF		user_private_message()	Sec. 5.1.11

wm_message_block_length – This 8-bit value shall specify the number of remaining bytes in the wm_message_block() that immediately follows this field up to and including the CRC_32 field.

wm_message_version – This 4-bit value shall be incremented if and only if anything in the watermark message changes, with wrap-around to 0 after the value reaches 15. The watermark processor in the receiving device is expected to use wm_message_version to discard duplicates. The video signal may include repeated instances of the same watermark message to improve reliability of delivery.

fragment_number – This 2-bit or 8-bit value shall specify the number of the current message fragment minus one. When (wm_message_id & 0x80)==0) , i.e. bit 7 is value '0', then fragment_number shall be 2 bits in length. When (wm_message_id & 0x80)==1) , i.e. bit 7 is value '1', then fragment_number shall be 8 bits in length. For example, value of 0 in fragment_number indicates the wm_message_block() carries the first (or only) fragment of a message, a fragment_number value of 1 indicates the wm_message_block() carries the second fragment of a message, etc. The value of fragment_number shall be less than or equal to the value of last_fragment.

last_fragment – This 2-bit or 8-bit value shall specify the fragment number of the last fragment used to deliver the complete watermark message. When (wm_message_id & 0x80)==0) , i.e. bit 7 is value '0', then last_fragment shall be 2 bits in length. When (wm_message_id & 0x80)==1) , i.e. bit 7 is value '1', then last_fragment shall be 8 bits in length. A value of zero in last_fragment indicates no segmentation is used (the watermark message contained within is complete). A value of 1 in last_fragment indicates the wm_message() will be delivered in two parts, a value of 2 indicates the watermark message will be delivered in three parts, and a value of 3 indicates it will be delivered in four parts, etc. The pair of values fragment_number and last_fragment may be considered to signal "part M of N".

wm_message_bytes() – When the value of last_fragment is 0, wm_message_bytes() shall be a complete instance of the watermark message identified by the value of wm_message_id. When the value of last_fragment is non-zero, wm_message_bytes() shall be a fragment of that watermark message. The concatenation of all instances of wm_message_block() with a given wm_message_id and

wm_message_fragment_version number shall result in the complete watermark message associated with that wm_message_id.

message_CRC_32 – When a message is sent in two or more fragments (e.g. last_fragment > 0) a 32-bit CRC covering the complete message (before segmentation) shall be provided in the last fragment of a fragmented message. The message_CRC_32 field shall not be present for non-fragmented messages (e.g. when the value of last_fragment is 0) or in any fragment other than the last (e.g. when fragment_number ≠ last_fragment). The message_CRC_32, when present, shall contain the CRC value that gives a zero output of the registers in the decoder defined in ISO/IEC 13818-1 [13], Annex A after processing the entire re-assembled message payload formed by concatenating the wm_message_id and wm_message_bytes(i) as specified in Table 5.4. The generating polynomial shall be $1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26}$.

CRC_32 – This 32-bit field shall contain the CRC value that gives a zero output of the registers in the decoder defined in ISO/IEC 13818-1 [13], Annex A after processing the entire message block. The generating polynomial shall be $1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26}$.

The wm_message_block() can deliver fragments of watermark messages that are intended to be reassembled before being processed further. The wm_message() data structure specified in Table 5.4 below represents the reassembled fragments. The definitions of wm_message_id, last_fragment, and message_CRC_32 shall be as specified above for wm_message_block(). The wm_message_bytes(i) field shall represent the wm_message_bytes() contained in the ith fragment of the message (counting from zero).

Table 5.4 Bit Stream Syntax of the Reassembled Watermark Message

Syntax	No. of Bits	Format
wm_message() {		
wm_message_id	8	uimsbf
for i=0; i<=last_fragment; i++) {		
wm_message_bytes(i)	var	Table 5.2
}		
message_CRC_32	32	uimsbf
}		

5.1.2 Multiplexing and Processing Rules

For the purposes of this specification, the term "short-form message" shall mean a watermark message using the encoding format with 2-bit fragment_number and last_fragment fields. The term "long-form message" shall mean a watermark message using the encoding format with 8-bit fragment_number and last_fragment fields. When constructing the sequence of watermark messages, the sending side shall conform to the following rules:

1) If a wm_message_block() of a given form (long or short) is sent with a fragment_number value of 0 (meaning first or only fragment) and a non-zero value of last_fragment (meaning that it will be sent in two or more fragments), each successive wm_message_block() of the same form shall have the same value in wm_message_id until the last fragment is sent (e.g. the value of fragment_number equals the value of last_fragment), with no intervening messages of the same form with other values of wm_message_id.

2) Fragments of any given message shall be sent in order. In other words, if n fragments are sent, the fragment_number values for any given message shall start at 0 and increase monotonically until the last fragment is sent. The fragment_number value in the last fragment shall be equal to the value of last_fragment.
3) Any given wm_message_block() (as indicated by the values of wm_message_id plus wm_message_version) may be sent multiple times. Receivers are expected to discard duplicates.

In accordance with the above rules, instances of message blocks comprising a long-form message may be interleaved with instances of message blocks comprising a short-form message. At most, fragments of only one short-form and one long-form message may be transmitted at any given time. A receiver can thus implement one buffer for short-form messages and a single separate buffer for long-form messages.

5.1.3 Segmentation and Reassembly Examples

A segmentation/reassembly mechanism allows the syntax of the watermark payload to accommodate the delivery of messages that exceed the size that will fit within one video frame. As noted, an "M of N" scheme is used. Figure 5.1 illustrates segmentation and reassembly for four different example messages, each of different lengths. Note the presence of the 32-bit message_CRC_32 in the last fragment of the segmented messages.

The first three messages in the examples in Figure 5.1 are the short-form variety (i.e., they can be delivered in four fragments or less). The fourth message has an 88-byte payload and thus uses the long-form syntax (which allows up to 256 fragments). Note that per the rules given in Section 5.1.2, fragments of the long-form message are interleaved with fragments of the short-form messages.

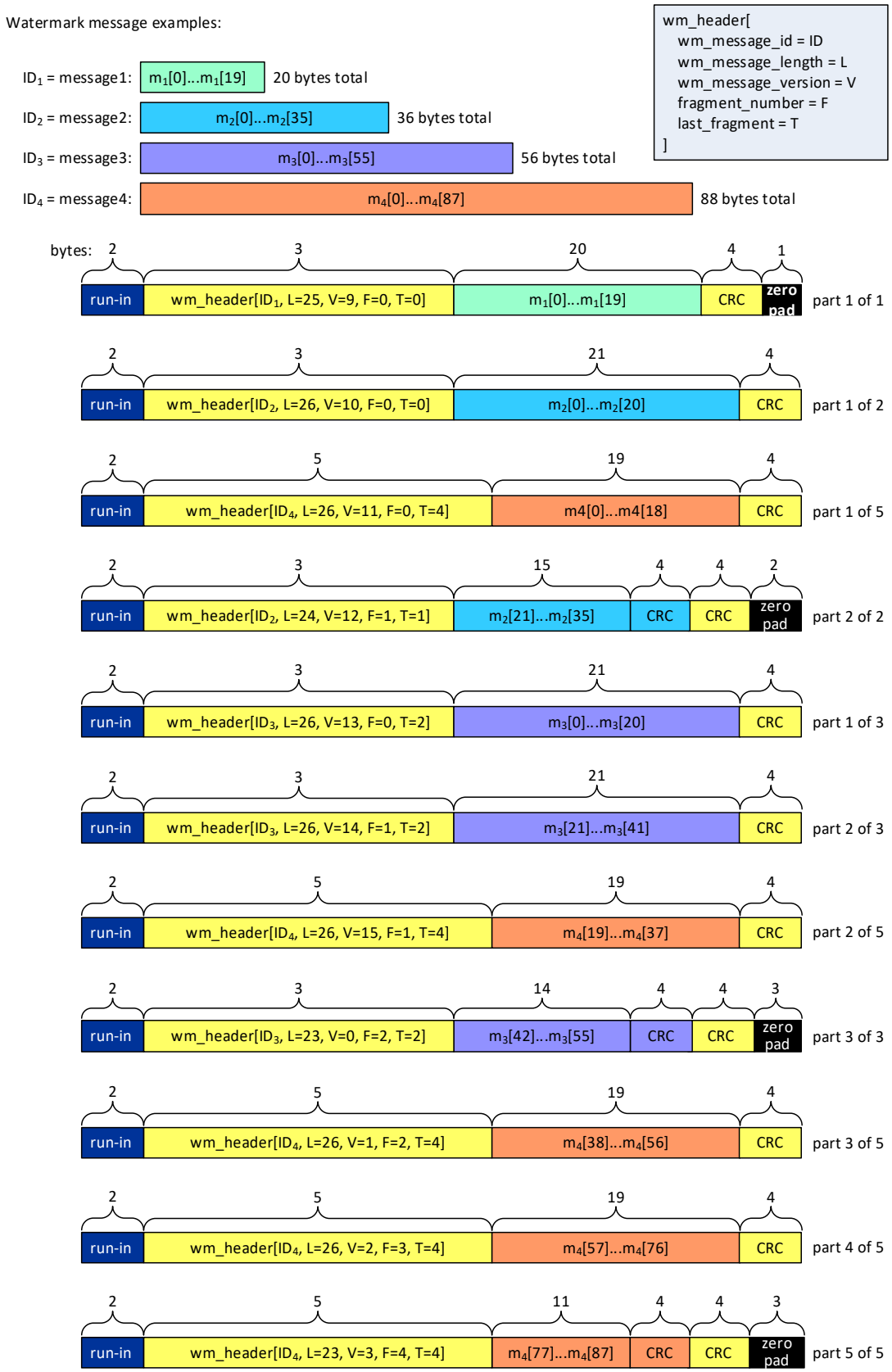

Figure 5.1 Illustration of message segmentation and interleaving (1X System).

5.1.4 Content ID Message

The Content ID Message can deliver a content identifier associated with the current program, and the major/minor channel number associated with the current Service. The Content ID Message may be sent in fragments (e.g. the value of last_fragment in the message header may be non-zero). Two types of content identifiers are currently defined, EIDR identifiers registered with the Entertainment Industry Data Registry (EIDR, (http://eidr.org/), and Ad-ID identifiers registered with the Ad-ID registry (http://ad-id.org) supported by the Association of National Advertisers (ANA) and the American Association of Advertising Agencies (4A's). There is also support for future content identifier systems that might be specified by ATSC and for additional experimental or proprietary Content ID systems. The syntax and bitstream semantics of the Content ID Message shall be as given in Table 5.5 and the parameter descriptions that follow.

Table 5.5 Bit Stream Syntax for the Content ID Message

Syntax	No. of Bits	Format
content_id_message() {		
content_ID_present	1	bslbf
channel_ID_present	1	bslbf
reserved	6	'111111'
if (content_ID_present) {		
reserved	1	'1'
valid_until_present	1	bslbf
content_ID_type	6	uimsbf
content_ID_length /* N1	8	uimsbf
If (valid_until_present=='1') {		
valid_until_time	32	uimsbf
reserved	6	'111111'
valid_until_time_ms	10	uimsbf
}		
if (content_ID_type==0x01) {		
EIDR	N1*8	uimsbf
}		
else if (content_ID_type==0x02) {		
adID_string()	N1*8	ASCII char. string
}		
else if (content_ID_type==0x3F) {		
private_ID_value	N1*8	Table 5.21
}		
else {		
reserved	N1*8	
}		
}		
if (channel_ID_present) {		
BSID	16	uimsbf
reserved	4	'1111'
major_channel_no	10	uimsbf
minor_channel_no	10	uimsbf
}		

```
}
```

content_ID_present – This one-bit Boolean flag shall indicate, when set to '1', that the fields associated with content ID are present in the message. When set to '0', the fields associated with content ID shall not be present.

channel_ID_present – This one-bit Boolean flag shall indicate, when set to '1', that the fields associated with channel ID are present in the message. When set to '0', the fields associated with channel ID shall not be present.

valid_until_present – This one-bit Boolean flag shall indicate, when set to '1', that the fields associated with valid_until_time and valid_until_time_ms are present in the message. When set to '0', the fields associated with valid_until_time and valid_until_time_ms shall not be present.

content_ID_type – This 5-bit unsigned integer field shall identify the type of content identifier provided in the message coded according to Table 5.6 below. Note that additional Content ID types might exist as indicated in the ATSC Code Point Registry [21].

Table 5.6 content_ID_type Field Encoding

content_ID_type Value	Meaning
0x00	ATSC Reserved
0x01	EIDR
0x02	Ad-ID
0x03–0x3E	Industry Reserved. See ATSC Code Point Registry [21].
0x3F	User Private Message. See private_ID_value below.

content_ID_length – This 8-bit unsigned integer shall signal the number of bytes in the EIDR identifier or Ad-ID identifier, private, or other value to follow. The content_ID_length value shall be 12 when the content_ID_type value is 0x01 (EIDR).

valid_until_time – This 32-bit unsigned integer shall indicate the end of the validity interval of the Content ID on the Recovery Media Timeline, encoded as the least-significant 32 bits of the count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI).

valid_until_time_ms – This 10-bit unsigned integer in the range 0 to 999 shall indicate the milliseconds offset from the time indicated in valid_until_time, such that the formula valid_until_time + (valid_until_time_ms/1000) yields the validity interval end time of the Content ID to the nearest 1 millisecond.

EIDR – This 96-bit (12 byte) value should be an identifier registered with EIDR (http://eidr.org/) and it shall be in the "compact binary" format as specified in SMPTE 2079 [15] Section 11.2. It shall be the EIDR identifier associated with the current broadcast content.

For example, given following EIDR in Canonical form:

`10.5240/7791-8534-2C23-9030-8610-5`, the 96-bit binary format would be: `0x1478779185342C2390308610`. The check digit is not included.

adID_string() – This ASCII character string shall represent an Ad-ID identifier formatted as an 11- or 12-character alphanumeric string per the Canonical Full Ad-ID Identifier Syntax specified in SMPTE RP 2092-1 [16] Section 4 or the ASCII characters representing the decimal value

of the 4-byte unsigned integer Compact Ad-ID Identifier as specified in SMPTE RP 2092-1 [16] Section 5. It shall be the Ad-ID identifier associated with the current broadcast content. The adID_string() should be a valid Ad-ID registered at http://www.ad-id.org

private_ID_value – This field shall represent the Content Identifier of a user private content identifier. The length of the ID in bytes shall be as given by the value of content_ID_length. The contents of this field shall conform to the User Private Message specified in Section 5.1.11.

BSID – This 16-bit unsigned integer field shall identify the Broadcast Stream ID of the original emission signal. In the case that the service is delivered via channel bonding at the physical layer, the indicated BSID shall be the lowest value among the set of BSID values representing the bonded RF emissions.

major_channel_no – This 10-bit unsigned integer field shall identify the Major Channel Number associated with the content. The combination of major and minor channel numbers shall be scoped to the broadcast area.

minor_channel_no – This 10-bit unsigned integer field shall identify the Minor Channel Number associated with the content.

The following constraints related to various fields in the Content ID message apply:

- At least one of content_ID_present or channel_ID_present shall have a value equal to '1'.
- When channel_ID_present is equal to 0, content_ID_present is equal to 1 and valid_until_present is equal to 1, the value of content_ID_length shall be less than or equal to 71 for 1X video watermark emission format (1X System) and shall be less than or equal to 191 for 2X video watermark emission format (2X System) [5]
- When channel_ID_present is equal to 0, content_ID_present is equal to 1 and valid_until_present is equal to 0, the value of content_ID_length shall be less than or equal to 77 for 1X video watermark emission format (1X System) and shall be less than or equal to 197 for 2X video watermark emission format (2X System) [5].
- When channel_ID_present is equal to 1, content_ID_present is equal to 1 and valid_until_present is equal to 1, the value of content_ID_length shall be less than or equal to 66 for 1X video watermark emission format (1X System) and shall be less than or equal to 186 for 2X video watermark emission format (2X System) [5].
- When channel_ID_present is equal to 1, content_ID_present is equal to 1 and valid_until_present is equal to 0, the value of content_ID_length shall be less than or equal to 72 for 1X video watermark emission format (1X System) and shall be less than or equal to 192 for 2X video watermark emission format (2X System) [5].

5.1.5 Presentation Time Message

A Presentation Time Message shall be carried within a single watermark payload. (I.e., the value of last_fragment shall be 0 for a message block carrying a Presentation Time Message.) The Presentation Time Message provides an indication to the receiver of the presentation time of the video frame carrying the watermark on the Recovery Media Timeline. If a given instance of the presentation_time_message() is repeated in immediate succession, the indicated presentation time shall correspond with the video frame of the first occurrence. The syntax and bitstream semantics of the presentation_time_message() shall be as given in Table 5.7 and the parameter descriptions that follow.

Table 5.7 Bit Stream Syntax for the Presentation Time Message

Syntax	No. of Bits	Format
presentation_time_message() {		
presentation_time	32	uimsbf
reserved	6	'111111'
presentation_time_ms	10	uimsbf
}		

presentation_time – This 32-bit unsigned integer shall indicate the presentation time of the frame associated with the watermark on the Recovery Media Timeline, encoded as the least-significant 32 bits of the count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI).

presentation_time_ms – This 10-bit unsigned integer in the range 0 to 999 shall indicate the milliseconds offset from the time indicated in presentation_time, such that the formula presentation_time + (presentation_time_ms/1000) yields the actual presentation time to the nearest 1 millisecond.

5.1.6 URI Message

The URI Message is used to deliver URIs of various types. The URI Message may be sent in fragments (e.g. the value of last_fragment in the message header may be non-zero).

Table 5.8 Bit Stream Syntax for the URI Message

Syntax	No. of Bits	Format
uri_message() {		
uri_type	8	uimsbf
domain_code	8	uimsbf
entity_strlen /* N1	8	uimsbf
entity_string()	8*N1	
uri_strlen /* N2	8	uimsbf
uri_string()	8*N2	
}		

uri_type – An 8-bit unsigned integer field that shall identify the type of URI to follow, according to the encoding given in Table 5.9. Note that additional encodings might exist as indicated in the ATSC Code Point Registry [21].

Table 5.9 uri_type field Encoding

uri_type	Meaning
0x00	ATSC Reserved
0x01	URL of Signaling Server (providing access to the Service Layer Signaling (SLS), as specified in Section 7 of A/331 [1]). Refer to A/331 [1] Section 6.7 for use.
0x02	URL of ESG data server (providing access to the ESG data, as specified in A/332 [2]). Refer to A/332 [2] Section 5.5.2 for use.
0x03	URL of Service Usage Data Gathering Report server (for use in reporting service usage, as specified in A/333 [3]). Refer to A/333 [3] Section 4.2.3 for use.
0x04	URL of dynamic event WebSocket server (providing access to dynamic events via WebSocket protocol, as specified in A/337 [6]). Refer to A/337 Section 5.5 for use.
0x05	URL of Advanced Emergency Alert Table (AEAT) associated with this Service. The XML syntax of the AEAT is defined in Section 6.5 of A/331 [1]. Refer to Section 5.4.5 below for use.
0x06-0xFF	Industry Reserved. See ATSC Code Point Registry [21].

domain_code – This 8-bit unsigned integer shall indicate the identifier code that shall identify the domain to be used for URL construction, according to Table 5.10.

Table 5.10 domain_code Encoding

domain_code value	domain_string()
0x00	`vp1.tv`
0x01 – 0xFF	ATSC Reserved

entity_strlen – An 8-bit unsigned integer that shall signal the number of characters in the entity_string() to follow.

entity_string() – This string shall be a portion of a RFC 3986 [9] URL, and shall consist only of Unreserved Characters (as defined in RFC 3986 [9] Sec 2.3), such that the URL conveyed by the uri_message() complies with RFC 3986. The length of entity_string() shall be as given by the value of entity_strlen.

uri_strlen – An 8-bit unsigned integer that shall signal the number of characters in the uri_string() to follow. The value of the uri_strlen field shall be less than or equal to 78 for 1X video watermark emission format (1X System) and shall be less than or equal to 198 for 2X video watermark emission format (2X System) [5].

uri_string() – This string shall be a portion of a RFC 3986 [10] URL, such that the URL conveyed by the uri_message() complies with RFC 3986. The length of the string shall be as given by the value of uri_strlen.

The URL signaled by the uri_message() shall be the concatenation of "`https://`", followed by a hostName determined from entity_string() and domain_string() using the method specified in Section 5.4.2, followed by "/" (forward slash), followed by uri_string(). This URL, after reassembly if sent in fragments, shall be a valid URL per RFC 3986 [10].

Accordingly, the URL is assembled as follows:

`https://`hostName/*uri_string()*

5.1.7 VP1 Message

The VP1 Message enables the recovery process (specified in Section 5.3) to be employed in conjunction with the Video Watermark.

A VP1 Video Watermark Segment shall consist of video content carrying a series of successive VP1 Message Groups whose initial video frames are nominally at 1.5 second intervals such that if the initial video frame of the first VP1 Message Group in a VP1 Video Watermark Segment occurs at time T seconds in the presentation, the initial video frame of the nth successive VP1 Message Group in the VP1 Video Watermark Segment occurs within ±0.5 video frames of time $T+1.5n$ seconds. All VP1 Message Groups in a VP1 Video Watermark Segment shall have the same Server Code and successive VP1 Message Groups in a VP1 Video Watermark Segment shall have sequentially incrementing Interval Codes. The query_flag value in the VP1 payload may change between successive VP1 Message Groups in a VP1 Video Watermark Segment.

When theVP1 Message is carried in a video component of audiovisual content for which an audio component employs an audio watermark carrying the same VP1 Payload, the VP1 Message Groups of the video watermark shall be time-aligned in the presentation such that the initial video frame in every VP1 Message Group occurs within ±0.5 video frames of the corresponding starting Cell boundary in the VP1 audio watermark.

Because the VP1 Message contains header and parity bits in addition to payload bits and because it is always repeated in multiple video frames, it may be recoverable from content for which the run-in sequence is not recoverable or where there are bit errors that cause the CRC-32 check to fail. Receivers may attempt to recover the VP1 Message in instances where run-in sequence recovery or CRC-32 check is unsuccessful.

5.1.7.1 vp1_message()

The bit stream syntax of the VP1 Message shall be as shown in Table 5.11.

Table 5.11 Bit Stream Syntax for the vp1_message()

Syntax	No. of Bits	Format
vp1_message() {		
header	32	bslbf
packet()	127	bslbf
zero_pad	1	‘0’
}		

header – This 32-bit field shall consist of a header element as specified in ATSC A/334 Audio Watermark Emission [4].

packet() – This 127-bit field shall be as given by Table 5.22 and the parameter descriptions that follow.

zero_pad – This one-bit field shall be set to value ‘0’.

When present, the vp1_message() shall be the first (i.e. left-most) wm_message() present in a video frame. When present, vp1_message()s carrying identical data shall be repeated for all successive video frames across at least a 1/6 second duration of content. The value of wm_message_version does not increment between vp1_message()s within a VP1 Message Group.

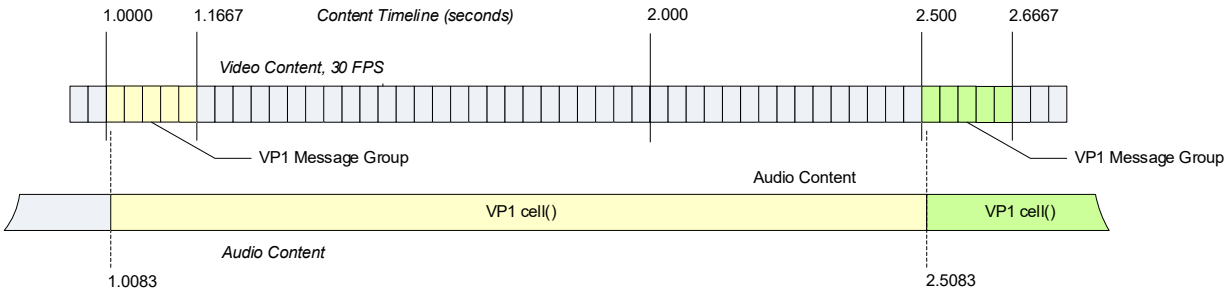

Figure 5.2 Temporal structure of VP1 Message Groups carrying vp1_message().

Figure 5.2 illustrates the temporal structure of VP1 Message Groups carrying a vp1_message() in a VP1 Video Watermark Segment, with time alignment in the presentation to a VP1 Audio Watermark Segment. The yellow-shaded VP1 Message Group carries the same VPI Payload as does the yellow-shaded VP1 cell() in the audio signal. Note that the VP1 Message Group spacing is exactly 1.5 seconds and the audio signal VP1 cell() is offset from the initial video frame of each VP1 Message Group by ¼ of a video frame period.

5.1.7.2 extended_vp1_message()

The bit stream syntax of the extended_vp1_message() shall be as shown in Table 5.12.

Table 5.12 Bit Stream Syntax for the extended_vp1_message()

Syntax	No. of Bits	Format
extended_vp1_message() {		
time_offset	8	uimsbf
header	32	bslbf
alternate_packet()	127	bslbf
zero_pad	1	‘0’
}		

time_offset – This 8-bit field shall convey the time offset of the video frame in which this extended_vp1_message() is carried relative to the first frame in its VP1 Message Group, in units of 1/30 of a second.

header – This 32-bit field shall consist of a header element as specified in ATSC A/334 Audio Watermark Emission [4].

alternate_packet() – This 127-bit field shall be as given by Table 5.22 and the parameter descriptions that follow; however, the alternate_parity_whitening_sequence and alternate_payload_whitening_sequence given in Table 5.13 shall be employed in place of the parity_whitening_sequence and payload_whitening_sequence given in Table 5.23.

zero_pad – This one-bit field shall be set to value ‘0’.

Table 5.13 Alternate Whitening Sequences

	alternate_parity_whitening_sequence	alternate_payload_whitening_sequence
Binary	001110011100101001111000111100000 111001010011001000110111101001000 10011111000	110010011101000111010111000100100110011 10101101100

When present, the extended_vp1_message() shall be the first (i.e., left-most) wm_message() present in a video frame.

Within a VP1 Message Group, an extended_vp1_message() shall be conveyed in at least those video frames whose sampling instant is within a right half-open time interval starting at a time that is an integer multiple of 0.3 seconds following the initial frame of the VP1 Message Group and ending 1/30 of a second later, for integers 0 through 4, unless a vp1_message() is also present in the VP1 Message Group, in which case for integers 1 through 4. When both vp1_message() and extended_vp1_message() are included in a VP1 Message Group, both message types shall be used in every VP1 Message Group of the VP1 Video Watermark Segment.

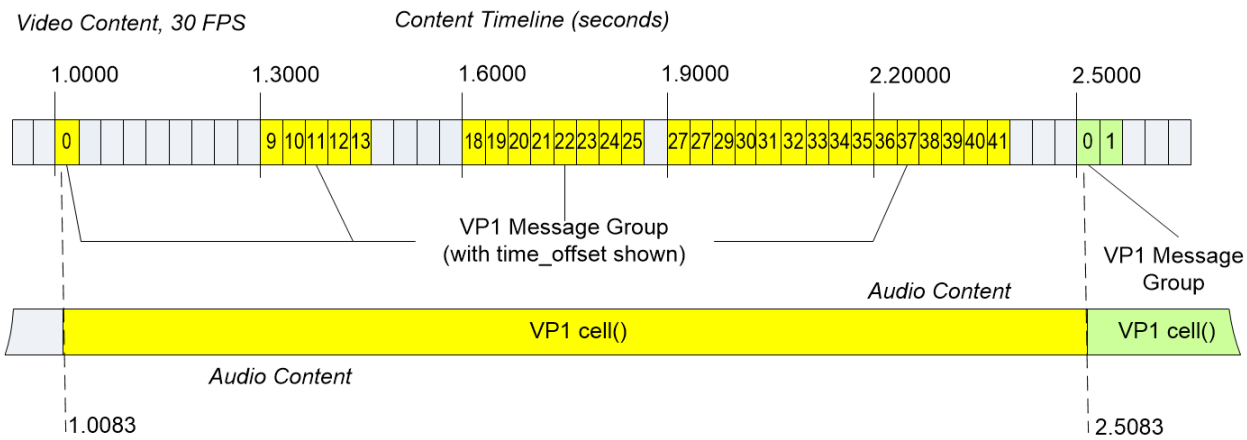

Figure 5.3 Temporal structure of VP1 Message Groups carrying extended_vp1_message().

Figure 5.3 illustrates the temporal structure of VP1 Message Groups carrying an extended_vp1_message() in a VP1 Video Watermark Segment, with time alignment to a VP1 Audio Watermark Segment. The yellow-shaded VP1 Message Group carries the same VP1 Payload as does the yellow-shaded VP1 cell() in the audio signal. Note that the VP1 Message Group spacing is exactly 1.5 seconds and the audio signal VP1 cell() is offset from the initial video frame of each VP1 Message Group by ¼ of a video frame period. Each video frame in the VP1 Message Group carries the same VP1 Payload as the aligned VP1 cell() and has a time_offset value equal to its offset from the first frame in the VP1 Message Group, since the frame rate is 30 fps.

Figure 5.4 illustrates the temporal structure of VP1 Message Groups carrying both vp1_message() and extended_vp1_message() in a VP1 Video Watermark Segment, with time alignment to a VP1 Audio Watermark Segment. The yellow-shaded VP1 Message Group carries the same VP1 Payload as does the yellow-shaded VP1 cell() in the audio signal. Note that the yellow-shaded video frames without time offset values convey vp1_message() and the yellow-shaded video frames with time offset values convey extended_vp1_message().

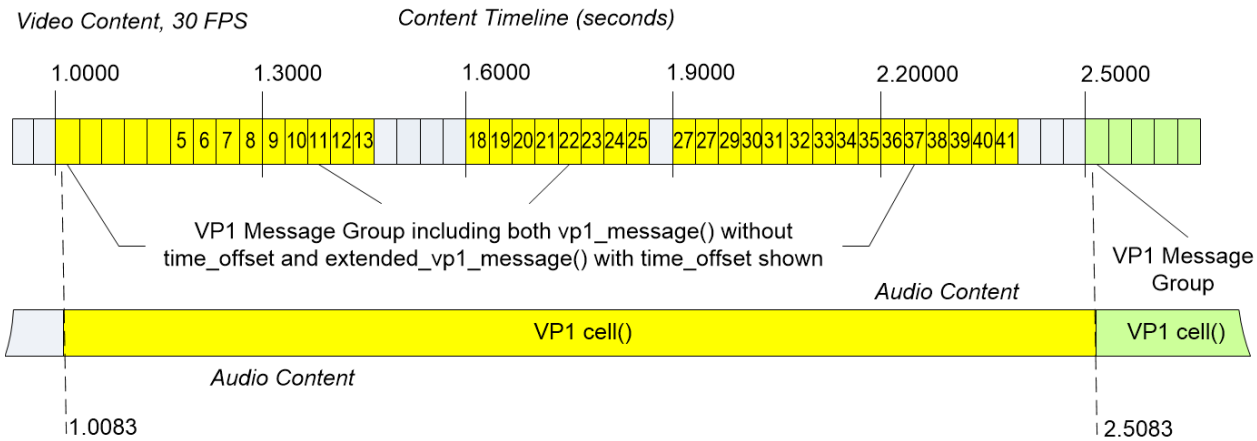

Figure 5.4 Temporal structure of VP1 Message Groups carrying both vp1_message() and extended_vp1_message().

5.1.8 Dynamic Event Message

The dynamic_event_message() supports delivery of Dynamic Events in video watermarks. The syntax and bitstream semantics of the Dynamic Event Message shall be as given in Table 5.14 and the parameter descriptions that follow.

Table 5.14 Bit Stream Syntax for the Dynamic Event Message

Syntax	No. of Bits	Format
dynamic_event_message() {		
delivery_protocol_type	4	uimsbf
reserved	4	'1111'
if (delivery_protocol_type == '1' || '2') {		
scheme_id_uri_length (N1)	8	uimsbf
scheme_id_uri_string	8*N1	
value_strlen (N2)	8	uimsbf
value_string	8*N2	
timescale	32	uimsbf
presentation_time	32	uimsbf
reserved	6	'111111'
presentation_time_ms	10	
duration	32	uimsbf
id	32	uimsbf
data_length (N3)	8	uimsbf
data	8*N3	
} else {		
reserved1_field_length (N1)	8	uimsbf
reserved1	8*N1	'11..'
}		
}		

delivery_protocol_type – This 4-bit field shall signify the delivery protocol (e.g., MMT or ROUTE/DASH) of the service to which the Dynamic Event applies. Table 5.15 below

describes the encoding of this field. Note that additional encodings might exist as indicated in the ATSC Code Point Registry [21].

Table 5.15 delivery_protocol_type field Encoding

delivery_protocol_type	Meaning
0	ATSC Reserved
1	ROUTE/DASH
2	MMTP
3-15	Industry Reserved. See ATSC Code Point Registry [21].

scheme_id_uri_strlen – This 8-bit unsigned integer field shall give the length of the scheme_id_uri_string field in bytes.

scheme_id_uri_string – This string shall give the `schemeIdUri` for the Event Stream of the Event.

value_strlen – This 8-bit unsigned integer field shall give the length of the value_string field in bytes.

value_string – This string shall give the `value` for the Event Stream of the Event.

timescale – This 32-bit unsigned integer shall give the time scale for the Event Stream of the Event, in ticks/second as defined in the MPEG DASH standard [14], to be used for the duration field.

presentation_time – This 32-bit unsigned integer shall indicate the presentation time of the Event on the Recovery Media Timeline, encoded as the least-significant 32 bits of the count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI).

presentation_time_ms – This 10-bit unsigned integer in the range 0 to 999 shall indicate the milliseconds offset from the time indicated in presentation_time, such that the formula presentation_time + (presentation_time_ms/1000) yields the actual presentation time to the nearest 1 millisecond.

duration – This 32-bit unsigned integer shall give the duration of the Event, in the time scale of the Event.

id – This 32-bit unsigned integer field shall be an ID for the Event, unique within the Event Stream.

data_length – This 8-bit integer shall give the length of the data field in bytes.

data – This field shall contain data needed for responding to the event, if any. The format and use of the data is determined by the Event Stream specification, which will be defined in the standard for any standards-based Event, and which will be known to any application registering to receive the Event for any Event targeted to applications.

reserved1_field_length – This 8-bit unsigned integer field shall give the length in bytes of the reserved1 field, which immediately follows this field.

reserved1 – This is reserved for future use.

The following constraints apply:

- When **delivery_protocol_type** has a value equal to 1 or 2 and the message is sent in a short-form message, the sum of the values of the **scheme_id_uri_length**, **value_strlen**, and **data_length** fields shall be less than or equal to 58 for 1X video watermark emission format (1X System) and shall be less than or equal to 178 for 2X video watermark emission format (2X System) [5].
- When **delivery_protocol_type** has a value equal to 1 or 2 and the message is sent in a long-form message, the sum of the values of the **scheme_id_uri_length**, **value_strlen**, and **data_length** fields shall be less than or equal to 4,838 for 1X video watermark emission format (1X

System) and shall be less than or equal to 12,518 for 2X video watermark emission format (2X System).

- When **delivery_protocol_type** has a value designated in Table 5.12 as "Reserved" or "Industry Reserved", and does not have an encoding defined in the ATSC Code Point Registry [21], and the message is sent in a short-form message, the value of **reserved1_field_length** shall be less than or equal to 78 for 1X video watermark emission format (1X System) and shall be less than or equal to 198 for 2X video watermark emission format (2X System) [5].
- When **delivery_protocol_type** has a value designated in Table 5.12 as "Reserved" or "Industry Reserved", and does not have an encoding defined in the ATSC Code Point Registry [21], and the message is sent in a long-form message, the value of **reserved1_field_length** shall be less than or equal to 4,858 for 1X video watermark emission format (1X System) and shall be less than or equal to 12,538 for 2X video watermark emission format (2X System).
- When **delivery_protocol_type** has a value designated in Table 5.12 as "Industry Reserved" and does have an encoding defined in the ATSC Code Point Registry [21], the **reserved1_field_length** and **reserved1** fields are replaced with encoding defined by the registrant.

5.1.9 Display Override Message

The Display Override Message provides an indication that, for a specified duration, the source broadcast audio is expected to be presented without modification and the source broadcast video is expected to be rendered without any overlaid graphics or other obstructions. (See Table 5.16.)

Table 5.16 Bit Stream Syntax for the Display Override Message

Syntax	No. of Bits	Format
display_override_message() {		
reserved	4	'1111'
override_duration	4	uimsbf
}		

override_duration – A 4-bit unsigned integer field, that when in the range 1–15 shall indicate the number of seconds the override should continue, following receipt of the message – unless superseded by a subsequent display override message. When the value of override_duration is '0', the override is immediately over.

5.1.10 Advanced Emergency Alert Message

The Advanced Emergency Alert Message supports delivery of emergency alert information in video watermarks similar to the Advanced Emergency Alert Table (AEAT) that can be delivered in over-the-air signaling (see A/331 [1] Section 6.5).

This Advanced Emergency Alert message type enables delivery of a subset of the information in the AEAT via video watermark so that emergency information may be delivered to receivers that are not receiving the over-the-air broadcast and that do not have a currently functioning broadband connection. Due to the payload capacity constraints of a video watermark, the location information that may be provided in the AEAT is not present in this message. Receivers are expected to interpret all alerts as if the receiver is located within the affected area. It should be noted that the full AEAT signaled in an over-the-air broadcast may be retrieved via broadband (by receivers that have a currently functioning broadband connection) using the URI Message method described in Section 5.1.6 or the VP1 Message method as described in Sections 5.1.7 and 5.2.

The syntax and bitstream semantics of the AEA_message() shall be as given in Table 5.17 and the parameter descriptions that follow.

Table 5.17 Bit Stream Syntax for the Advanced Emergency Alert Message

Syntax	No. of Bits	Format
AEA_message() {		
AEA_id_length_minus1 (N1)	5	uimsbf
AEA_type	3	uimsbf
priority	3	uimsbf
AEA_issuer_length_minus1 (N2)	5	uimsbf
AEA_id	8*(N1+1)	(N1+1)*char
AEA_issuer	8*(N2+1)	(N2+1)*char
audience	3	uimsbf
ref_AEA_id_present_flag	1	bslbf
AEA_wakeup_flag	1	bslbf
AEAT_url_present_flag	1	bslbf
reserved	2	‘11’
num_AEA_text_minus1	2	uimsbf
num_eventDesc	2	uimsbf
reserved	4	‘1111’
effective	32	uimsbf
expires	32	uimsbf
if (AEAT_url_present_flag == 1) {		
domain_code	8	uimsbf
entity_strlen_minus1 (N3)	8	uimsbf
entity_string()	8*(N3+1)	(N3+1)*char
AEAT_url_strlen_minus1 (N4)	8	uimsbf
AEAT_url_string	8*(N4+1)	(N4+1)*char
}		
if (ref_AEA_id_present_flag == 1) {		
ref_AEA_id_length_minus1 (N5)	8	uimsbf
ref_AEA_id	8*(N5+1)	(N5+1)*char
}		
for (i=0; i<num_eventDesc; i++) {		
eventDesc_length_minus1 (N6)	6	Uimsbf
reserved	2	‘11’
eventDesc	8*(N6+1)	(N6+1)*char
reserved	4	‘1111’
eventDesc_lang_length_minus1 (N7)	4	Uimsbf
eventDesc_lang	8*(N7+1)	(N7+1)*char
}		
for (i=0; i<num_AEA_text_minus1+1; i++) {		
reserved	4	‘1111’
AEA_text_lang_length_minus1 (N8)	4	Uimsbf
AEA_text_lang	8*(N8+1)	(N8+1)*char
AEA_text_length_minus1 (N9)	8	Uimsbf
AEA_text	8*(N9+1)	(N9+1)*char
}		

}		

AEA_id_length_minus1 – This 5-bit unsigned integer field plus 1 shall indicate the length of the AEA_id field in bytes.

AEA_type – This 3-bit unsigned integer field shall indicate the category of the AEA message. This unsigned integer shall be the value of the **AEAT.AEA**@AEAtype attribute of the current Advanced Emergency Alert message defined in [1]. The value shall be coded according to Table 5.18.

Table 5.18 Code Values for AEA_type Field

Code Value	AEA_type	Meaning
0x00	undefined	
0x01	Alert	Indicates that AEA message is new. (Note: alert messages such as the U.S. required monthly test, RMT, are considered alert messages, and AEA_type must be set to the value of 0x01).
0x02	Update	Indicates that AEA message is not new, but contains updated information from any previous emergency alert message.
0x03	Cancel	Indicates that AEA message is cancelling any previous emergency alert message, even when the message is not expired.
0x04-0x07	other values	ATSC Reserved for future use

priority – This 3-bit unsigned integer shall be the value of the **AEAT.AEA**@priority attribute of the current Advanced Emergency Alert message defined in [1].

AEA_issuer_length_minus1 – This 5-bit unsigned integer field plus 1 shall indicate the length of the AEA_issuer field in bytes.

AEA_id – This string shall be the value of the **AEAT.AEA**@AEAid attribute of the current Advanced Emergency Alert message defined in [1].

AEA_issuer – This string shall be the value of the **AEAT.AEA**@issuer attribute of the current Advanced Emergency Alert message defined in [1].

audience – This 3-bit unsigned integer field shall indicate the audience type of the message. This unsigned integer shall be the value of the **AEAT.AEA**@audience attribute of the current Advanced Emergency Alert message defined in [1]. The value shall be coded according to Table 5.19.

Table 5.19 Code Values for audience Field

Code Value	Audience	Meaning
0x00	Undefined	
0x01	Public	For general dissemination to unrestricted audiences. All alerts intended for public consumption must have the value of “public.” (Required for AEA public dissemination)
0x02	Restricted	For dissemination only to an audience with a defined operational requirement. Alerts intended for non-public dissemination may include the value of “restricted”.
0x03	Private	For dissemination only to specified addresses (conditional access requirement).
0x04-0x07	other values	ATSC Reserved for future use

ref_AEA_ID_present_flag – This 1-bit Boolean flag field shall indicate, when set to ‘1’, the presence of the ref_AEA_ID and ref_AEA_ID_length_minus1 fields in the AEA message. When ref_AEA_id_present_flag is set to ‘0’, these fields shall not be present. When the value of AEA_type is 0x01, ref_AEA_id_present_flag shall be set to ‘0’. When the value of AEA_type is 0x02 or 0x03, ref_AEA_id_present_flag shall be set to ‘1’.

AEA_wakeup_flag – This 1-bit Boolean flag field shall be the value of the optional **AEAT.AEA**@wakeup attribute defined in [1]. When the **AEAT.AEA**@wakeup attribute is not present, this field shall be set to '0'.

AEAT_url_present_flag – This 1-bit Boolean flag field shall indicate, when set to '1', the presence of the domain_code, entity_strlen_minus1, entity_string, AEAT_url_strlen_minus1, and AEAT_url_string fields in the AEA message. When AEAT_url_present_flag is set to '0', these fields shall not be present.

num_AEA_text_minus1 – This 2-bit unsigned integer field plus 1 shall indicate the number of the AEA_text fields present in the AEA message. The value of num_AEA_text_minus1 shall not be greater than 2.

num_eventDesc – This 2-bit unsigned integer field shall indicate the number of event descriptions included in the AEA message, which shall not be greater than 2, where each event description consists of the fields eventDesc_length_minus1, eventDesc, eventDesc_lang_length_minus1, and eventDesc_lang, and the intervening reserved fields. The value of 0x03 shall be reserved for future use.

effective – This parameter shall indicate the effective date and time of AEA Message on the wall-clock timeline, encoded as a 32-bit count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI). This parameter shall be the value of the **AEAT.AEA.Header**@effective attribute of the current Advanced Emergency Alert message defined in [1].

expires – This parameter shall indicate the latest expiration date and time of AEA Message on the wall-clock timeline, encoded as a 32-bit count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI). This parameter shall be the value of the **AEAT.AEA.Header**@expires attribute of the current Advanced Emergency Alert message defined in [1].

domain_code – This 8-bit unsigned integer shall indicate the string that shall identify the domain to be used for URL construction, according to Table 5.20.

Table 5.20 domain_code Encoding

domain_code value	domain_string()
0x00	vp1.tv
0x01 – 0xFF	ATSC Reserved

entity_strlen_minus1 – This 5-bit unsigned integer plus 1 shall indicate the length of the entity_string() field in bytes.

entity_string() – This string shall be a portion of a RFC 3986 [9] URL, and shall consist only of Unreserved Characters (as defined in RFC 3986 [9] Sec 2.3), such that the URL the receiver may use to download the XML-formatted AEAT as defined in [1], defined below, complies with RFC 3986.

AEAT_url_strlen_minus1 – This 8-bit unsigned integer field plus 1 shall indicate the length of the AEAT_url_string field in bytes.

AEAT_url_string – This string shall be a portion of a RFC 3986 [9] URL, such that the URL the receiver may use to download the XML-formatted AEAT as defined in [1], defined below, complies with RFC 3986.

The URL that the receiver may use to download the XML-formatted AEAT as defined in [1] shall be the concatenation of "https://", followed by entity_string(), followed by "." (period),

followed by domain_string(), followed by "/" (forward slash), followed by AEAT_url_string(). This URL, after reassembly if sent in fragments, shall be a valid URL per RFC 3986 [9]. Accordingly, the URL is assembled as follows:

`https://`*entity_string()*`.`*domain_string()*`/`*AEAT_url_string()*

ref_AEA_ID_length_minus1 – This 8-bit unsigned integer field plus 1 shall indicate the length of the ref_AEA_ID field in bytes.

ref_AEA_ID – This string shall be the value of the **`AEAT.AEA`**`@refAEAid` attribute of the current Advanced Emergency Alert message defined in [1].

eventDesc_length_minus1 – This 6-bit unsigned integer plus 1 shall indicate the length of the eventDesc field in bytes.

eventDesc – This string shall be the value of the **`AEAT.AEA.Header.eventDesc`** character string of the current Advanced Emergency Alert message defined in [1].

eventDesc_lang_length_minus1 – This 4-bit unsigned integer field plus 1 shall indicate the length of the eventDesc_lang field in bytes.

eventDesc_lang – This string shall be the **`AEAT.AEA.Header.eventDesc`**`@lang` attribute of the current Advanced Emergency Alert message defined in [1].

AEA_text_lang_length_minus1 – This 4-bit unsigned integer field plus 1 shall indicate the length of the AEA_text_lang field in bytes.

AEA_text_lang – This string shall be the **`AEAT.AEA.AEAtext`**`@lang` attribute of the current Advanced Emergency Alert message defined in [1].

AEA_text_length_minus1 – This 8-bit unsigned integer field plus 1 shall indicate the length of the AEA_text field in bytes.

AEA_text – This string shall be the UTF-8 character encoded value of the **`AEAT.AEA.AEAtext`** element of the current Advanced Emergency Alert message defined in [1].

5.1.11 User Private Message

The User Private Message supports delivery of opaque user data in a managed manner. The syntax and bitstream semantics of the user_private_message() shall be as given in the parameter descriptions that follow (Table 5.21).

Table 5.21 Bit Stream Syntax for the User Private Message

Syntax	No. of Bits	Format
user_private_message() {		
domain_length_minus1 (N1)	8	uimsbf
domain	8*(N1)	
if (wm_message_id==0x01 || wm_message_id==0x7F) {		
payload_length_minus1 (N2)	8	uimsbf
}		
else {		
payload_length_minus1 (N2)	14	uimsbf
reserved	2	uimsbf
}		
payload	8*(N2)	
}		

domain_length_minus1 – This 8-bit unsigned integer field plus 1 shall indicate the length of the domain field in bytes.

domain – This variable length field shall conform to `taggingEntity` token as defined in Section 2.1 of IETF RFC 4151 [11], e.g. "`atsc.org,2016`". The registrant of `authorityName` in the `taggingEntity` token shall define the contents of the payload field.

payload_length_minus1 – The value of this field plus1 shall indicate the length of the payload field in bytes.

This field is an 8-bit unsigned integer when this user_private_message() is included as private_ID_value field in content_id_message() (wm_message_id equal to 0x01) or is signaled within a wm_message_block() with wm_message_id equal to 0x07F.

This field is a 14-bit unsigned integer when this user_private_message() is signaled within a wm_message_block() with wm_message_id equal to 0x0FF.

payload – This variable length field contains data not defined in this specification.

5.2 Audio Watermark Payload Format

The emission format for audio watermarks shall conform to the ATSC A/334 Audio Watermark Emission specification [4].

5.2.1 Packet

The bit stream syntax of the packet() structure shall be as shown below in Table 5.22. It is divided into two regions: scrambled Parity and scrambled VP1 Payload bits.

Table 5.22 Syntax of packet() Structure

Syntax	No. of Bits	Format
packet() {		
scrambled_parity	77	uimsbf
scrambled_vp1_payload	50	uimsbf
}		

scrambled_parity – The scrambled Parity bits, where Parity bits shall be as specified in Section 5.2.2.1, below.

scrambled_vp1_payload – The scrambled VP1 Payload, as specified in Section 5.2.2, below.

5.2.2 Scrambling

The scrambled_parity shall be the Parity bits scrambled via bitwise exclusive OR operation with the parity_whitening_sequence, shown below in Table 5.23. The scrambled_vp1_payload shall be the VP1 Payload bits, as specified in 5.2.4, scrambled via bitwise exclusive OR operation with the payload_whitening_sequence, shown below in Table 5.23.

Table 5.23 Whitening Sequences

	parity_whitening_sequence	payload_whitening_sequence
Binary	111001101111111110110110101111011 0010001000010010111000010010000 0 001101100101	0010000100001010001100000000101110000001 1100110111

5.2.2.1 Parity Bits

A BCH (127, 50, 13) code shall be applied to the VP1 Payload. We shall define the payload polynomial, $P(x)$, as the polynomial that results when the 50 payload bits are used to form a polynomial with the leftmost bit being the coefficient for order 49 and the rightmost bit being the coefficient for order 0. We shall define the extended payload polynomial, ($x^{77}P(x)$) as the payload polynomial $P(x)$, multiplied by x^{77}. This is equivalent to the polynomial whose coefficients are the payload bits appended (to the right) with 77 zero bits.

We shall define the generator polynomial $G(x)$ for the BCH (127, 50, 13) code as specified in in Table 5.24 below.

We shall define the remainder polynomial $R(x)$ as the remainder resulting from dividing the extended payload polynomial by the generator polynomial, or:

$$R(x) = (x^{77}P(x)) \bmod G(x)$$

The order of the remainder polynomial will be one less than that of the generator polynomial.

The Parity bits shall have the value of the coefficients of the remainder polynomial $R(x)$, where the leftmost Parity bit is the order 76 coefficient and the rightmost Parity bit is the order 0 coefficient. See Table 5.24.

Table 5.24 Polynomials used for BCH code (127, 50, 13)

Primitive Polynomial	Generator Polynomial
$x^7 + x^6 + 1$	$x^{77} + x^{76} + x^{75} + x^{74} + x^{72} + x^{71} + x^{68} + x^{67} + x^{66} + x^{64} + x^{63} + x^{62} + x^{60} + x^{59} + x^{51} + x^{50} + x^{49} + x^{44} + x^{42} + x^{41} + x^{40} + x^{39} + x^{35} + x^{34} + x^{32} + x^{30} + x^{29} + x^{26} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{13} + x^{12} + x^{9} + x^{5} + x^{2} + 1$

5.2.3 VP1 Payload

The syntax of the vp1_payload() structure is shown below in Table 5.25, Table 5.26 and Table 5.27.

Table 5.25 Syntax of payload() Structure

Syntax	No. of Bits	Format
vp1_payload() {		
domain_type	1	bslbf
If (domain_type == 0) {		
small_domain()	var	Table 5.26
}		
else {		
large_domain()	var	Table 5.27
}		
}		

Table 5.26 Syntax small_domain() Structure

Syntax	No. of Bits	Format
small_domain() {		
server_field	31	uimsbf
Interval_field	17	uimsbf
query_flag	1	bslbf
}		

Table 5.27 Syntax large_domain() Structure

Syntax	No. of Bits	Format
large_domain() {		
server_field	23	uimsbf
Interval_field	25	uimsbf
query_flag	1	bslbf
}		

domain_type – This 1-bit value specifies whether the payload information is for a small_domain() or a large_domain().

server_field – This field shall contain the Server Code. Although information about management of Server Codes is out of scope of this document, the Server Code value shall be globally unique and shall be acquired according to information found at https://www.atsc.org/336-server-code-registrar-information/.

interval_field – This field contains the Interval Code.

query_flag – This field signals when a Dynamic Event is available. A change in the value of this field between successive VP1 Payloads in a VP1 Audio Watermark Segment or between successive VP1 Message Groups in a VP1 Video Watermark Segment indicates that a Dynamic Event (as defined in A/337 [6]) is available from the Dynamic Event HTTP server.

The range of the Server Code and Interval Code as a function of domain type is shown below in Table 5.28.

Table 5.28 Supported Range of Server Codes and Interval Codes per Domain Type

Domain Type	Server Code Range (hexadecimal)	Interval Code Range (hexadecimal)
Small Domain	`00000000-7FFFFFFF`	`00000000-0001FFFF`
Large Domain	`00000000-007FFFFF`	`00000000-01FFFFFF`

5.2.4 Display Override State

The display override state may be signaled in the audio watermark to indicate that receivers should suspend any modification (audio or video) to the presentation of the marked content which is occurring as a result of signaling obtained via the recovery process.

The display override state shall be indicated by the use of inverse signaling in the audio watermark emission as defined in A/334 [4].

The display override state is in effect solely for the duration of the marked content where inverse signaling is employed.

Transitions between standard signaling and inverse signaling shall occur at symbol boundaries and are not required to occur at VP1 Audio Watermark Segment, Cell, header, Packet, or payload field boundaries.

5.2.5 VP1 Audio Watermark Segment

A VP1 Audio Watermark Segment is a continuously marked interval of content which contains VP1 payloads in contiguous Cells with identical Server Codes and sequentially incrementing Interval Codes.

The Query Flag value may change between contiguous VP1 payloads in a VP1 Audio Watermark Segment.

5.2.6 Examples (Informative)

As an aid to calculating the specified BCH(127,50,13) code parity bits, example bit sequences for valid cell() structures are shown below in Table 5.29. Note that the concatenation of the parity and payload form a valid BCH(127,50,13) codeword.

Table 5.29 Example cell() Sequences, Shown as Hexadecimal Values.

Header	Parity	VP1 Payload	Scrambled Parity	Scrambled Payload
AE0AB9E4	00000000000000000000	0000000000000	39BFE6D7B2212E120365	08428C02E0737
AE0AB9E4	1D9DD80E178D643E3225	0000000000001	01422ED9A5AC4A2C3140	08428C02E0736
AE0AB9E4	0CD1D8526D369D4A6D8E	1004B5A1C3B7F	100E2E85DF17B3586EEB	184639A323C48

5.3 Fingerprint Methodology

5.3.1 Introduction and Architecture

Two different models are commonly used for communication between remote fingerprint servers and receivers: a request/response model and an event-driven model.

In the request/response model, the receiver is expected to compute signatures of the content periodically (e.g. every few seconds) and send requests containing the signatures to a fingerprint server. When a fingerprint server gets a request from a receiver, it returns a response. The communications session is not kept open between request/response instances. In this model, it is not feasible for a fingerprint server to initiate messages to a receiver.

In the event driven model, the receiver is expected to initiate a persistent connection to the fingerprint server, compute signatures of frames periodically (e.g., every few seconds), and submit the signatures over the connection. In this scenario, the receiver includes a submission sequence number in the message with each submission. The fingerprint server does not respond to each submission. It sends a message to the receiver only when a new Recovery File or Dynamic Event is detected. In this model, it is possible for the fingerprint server to initiate messages to the client at any time.

5.3.2 Request/Response Model

A fingerprint server that is using the request/response model shall respond to each request from a receiver.

The response shall include the following parameters:

- Recovery Base URL (the URL scheme shall be `https:`)
- Media timing information, i.e., `intervalCode`

The response may include either or both of the following parameters:

- Dynamic Event Base URL (the URL scheme shall be `https:`)
- Dynamic Event WebSocket Server URL (the URL scheme shall be `wss:`)

The recovery base URL and `intervalCode` are used for a receiver to request a Recovery File from a recovery server. It is described in Section 5.4.1 how to construct a URL for a request of a Recovery File.

There are two ways that a receiver is able to retrieve Dynamic Events in the request/response model. If a Dynamic Event base URL is returned, a receiver is able to request a Dynamic Event from an event server. It is described in Section 5.4.4 how to construct a URL for a request of a Dynamic Event. If a Dynamic Event WebSocket server URL is returned, a receiver is able to keep a persistent connection to an event server via the WebSocket protocol. The detailed mechanism to receive Dynamic Events through the WebSocket connection shall be as described in Section 5.5 of [6].

5.3.3 Event-Driven Model

A fingerprint server that is using the event-driven model need not respond to each request from a receiver but shall respond when a new Recovery File or Dynamic Event is detected.

When a new Recovery File is detected, the response shall include following parameters:

- Recovery Base URL (the URL scheme shall be `https:`)
- Media timing information, i.e., `intervalCode`

When a new Dynamic Event is detected, the response shall include the following parameter:

- Dynamic Event Base URL (the URL scheme shall be `https:`)

The recovery base URL and `intervalCode` are used for a receiver to request a Recovery File from a recovery server. It is described in Section 5.4.1 how to construct a URL for a request of a Recovery File.

If a Dynamic Event base URL is returned, a receiver is able to request a Dynamic Event from an event server. It is described in Section 5.4.4 how to construct a URL for a request of a Dynamic Event.

5.4 Recovery Process

A “redistribution setting” is a situation in which a device is receiving audio/video TV content via an input, such as an HDMI cable, without direct access to the ATSC broadcast stream from which the content is derived (e.g., a device getting TV content from a cable, satellite or IPTV set-top box). The “recovery process” is the process that a device in a redistribution setting goes through to get broadband access to supplementary content provided by the broadcaster of the content that the device is receiving via the cable.

There is one recovery process that a device in a redistribution setting can go through if it is receiving ATSC 3.0 audio watermarks in the content it is receiving, if it is receiving VP1 messages in video watermarks, or if it is using the fingerprint method. This process is outlined in Annex D. Sections 5.4.1 through Section 5.4.4 provide details about this process.

There is a different recovery process that such a device can go through if it is receiving a full set of ATSC 3.0 video watermarks in the content it is receiving. This process is outlined in Annex C.

The use of the DNS protocol in connection with the recovery process shall conform to the requirements of A/360 [7]. HTTPS and WebSocket protocols shall employ Secure Connections as specified in A/360 [7].

5.4.1 Recovery File Retrieval via Broadband

A recovery data request can be made to a Recovery File Server by a receiver via issuance of a HTTPS GET request for a Recovery File.

When a receiver is receiving VP1 messages in audio watermarks or video watermarks, the Recovery File URL shall be constructed from a VP1 payload using the URL template:

```
https://{hostName}/a336/rdt/{subdName}/{serverCode}-
{intervalCode}.rdt[?query]
```

where:

`[string]` items in square brackets [..] indicate an optional `string` and

`{element}` items in curly brackets {..} indicate a named `element`.

The request shall use the prefix `https`. Note that future versions of this standard might define a query string (see RFC 3986 [10]). Section 5.4.2 specifies the method for determining `hostName`.

If the domain_type of the VP1 payload is '0', then `subdName` shall have the value:

```
{serverCode4}{serverCode3}/{serverCode2}/{serverCode1}
```

`serverCode` shall have the value:

```
{serverCode4}{serverCode3}{serverCode2}{serverCode1}
```

and if the domain_type of the VP1 watermark code is '1', then `subdName` shall have the value:

```
{serverCode3}{serverCode2}/{serverCode1}
```

and `serverCode` shall have the value:

```
{serverCode3}{serverCode2}{serverCode1}
```

where `serverCode1`, `serverCode2`, `serverCode3`, and `serverCode4` shall respectively mean the least-to-most significant bytes of the server_field of the VP1 payload (zero-padded at the most-significant bit to the next byte boundary) expressed as two-character uppercase-only hexadecimal values.

`intervalCode` shall be the value of the interval_field of the VP1 payload (zero-padded at the most-significant bit to the next byte boundary) expressed as an uppercase-only hexadecimal value (6 characters in length for VP1 payloads with domain_type=0 and 8 characters in length for VP1 payloads with domain_type=1).

When the fingerprint method is used, the Recovery File URL shall be constructed from a response of a fingerprint server using the URL template:

```
{Recovery Base URL}/{intervalCode}.rdt[?query]
```

where:

`[string]` items in square brackets [..] indicate an optional `string` and

`{element}` items in curly brackets {..} indicate a named `element`.

The response to a recovery data request shall contain a Recovery File as set forth in Section 5.4.3 packaged together with zero or more other signaling files delivered in the same response

body format defined for svcInetUrl HTTP requests in Section 6.7 of A/331 [1]. Recovery Files need not be signed. The "validFrom", "validUntil", and "availAt" attributes, if present, correspond to UTC times on the Recovery Media Timeline. Receivers are not expected to use or cache signaling files that are not valid (i.e., if validFrom is in the future or validUntil is in the past compared to the current playback position on the Recovery Media Timeline) at the time of their receipt. Note that future versions of this standard might define a query string (see RFC 3986 [10]).

5.4.2 Hostname Determination

The `hostName` to be used in video watermark URLs and in recovery and dynamic event requests shall be determined via DNS resolution of an intermediate name (`intName`).

For video watermark URLs, `intName` shall be the concatenation of entity_string(), followed by "." (period), followed by domain_string() as defined in Section 5.1.6.

For recovery and dynamic event requests, if the domain_type of the VP1 payload is '0', then `intName` shall have the value:

```
a336.{serverCode1}.{serverCode2}.{serverCode3}.{serverCode4}.0.vp1.tv
```

and if the domain_type of the VP1 watermark code is '1', then `intName` shall have the value:

```
a336.{serverCode1}.{serverCode2}.{serverCode3}.1.vp1.tv
```

where `{serverCode1}`, `{serverCode2}`, `{serverCode3}`, and `{serverCode4}` have the meanings given in Section 5.4.1.

The value of `hostName` shall be the canonical name that is given in the first CNAME record (as defined in RFC 1035 [9]) obtained during DNS resolution of the intermediate name `intName`. If DNS resolution of `intName` does not include any CNAME records, then `hostName` shall be equal to `intName`.

In any case, if `hostName` resolves to the unspecified IP address (0.0.0.0/32 in the case of an IPv4 address or ::/128 in the case of an IPv6 address), it indicates that network services are not supported for the given `hostName`. Network requests should not be issued to the unspecified address.

5.4.3 Recovery File Format

The Recovery File shall be a JSON document [17]. RDT instance documents shall use UTF-8 encoding. The normative JSON schema for Recovery File format shall be as specified in Annex B. Broadband delivery of Recovery Files shall employ the media type definition specified in Annex E.1. Informative Table 5.30 describes the structure of the Recovery File format in a more illustrative way. The specification following the table gives the normative semantics.

Note that each VP1 payload corresponds to one and only one Recovery File.

Table 5.30 Recovery File Format Logical Structure

Element or Attribute Name	Use	Data Type	Format
`RecoveryDataTable`	1		Root element
`thisComponent`	1		The media component from which the VP1 payload carrying `serverCode` and `intervalCode` was detected.
`serverCode`	0..1	integer	The `serverCode` value from the query request to which the recovery data table was provided as a response.

Element or Attribute Name	Use	Data Type	Format
intervalCode	0..1	integer	The intervalCode value from the query request to which the recovery data table was provided as a response.
queryFlag	0..1	integer	An integer number having a value of 0 or 1 matching the query_flag value in the VP1 payload associated with this RecoveryDataTable .
displayOverride	0..1	integer	An integer number having a value of 0 or 1 indicating whether a display override state was active at any part of the Cell related to the VP1 payload associated with this RecoveryDataTable.
componentDescription	1	object	The description of this component, as defined in Table 5.31.
querySpread	0..1	integer	Time window duration in milliseconds over which a device should randomly choose a time for querying Dynamic Event Server when signaled by the query_flag in the VP1 payload.
otherComponent	0..N	object	The audio or video components other than this component, which carry coincident VP1 payloads.
contentID	0..N		Content identifier.
type	1	string	Content ID type.
cid	1	string	Encoded identifier value.
validFrom	1	string (date-time)	When contentID is valid from
validUntil	0..1	string(date-time)	Up to when contentID is valid until
sourceID	0..1		
country	1	string	ISO 3166-1 alpha-2 [12] country code associated with the primary administrative entity under which the given bsid is assigned.
bsid	1	integer	Identifier of the whole Broadcast Stream.
majorChannelNo	1	integer	An integer number in the range 1 to 999 representing the “major” channel number of the service.
minorChannelNo	1	integer	An integer number in the range 1 to 999 representing the “minor” channel number of the service.
service	1		
serviceId	1	integer	Number that uniquely identifies this service within the scope of bsid.
sltSvcSeqNum	1	integer	Version of service information in this table – increments by one for each new version of service data in RDT; wraps around to 0.
slsProtocol	0..1	integer	Protocol used to deliver the service layer signaling for this service (See Table 5.15).
slsMajorProtocolVersion	0..1	integer	Major version number of the protocol used to deliver the service layer signaling for this service.
slsMinorProtocolVersion	0..1	integer	Minor version number of the protocol used to deliver the service layer signaling for this service.

Element or Attribute Name	Use	Data Type	Format
globalServiceID	1	string	A globally unique URI that identifies the currently selected service.
svcInetUrl	0..N		
urlType	1	unsignedByte	Type of files available with this URL (See Table 5.9)
urlValue	1	string (uri)	URL to access Internet signaling files for this service.

thisComponent – A description of the media component embedded with a watermark containing the VP1 payload containing serverCode and intervalCode.

serverCode – When present, this element shall provide the serverCode value employed in the HTTP request to which this Recovery File was provided as a response.

intervalCode – When present, this element shall provide the intervalCode value from the query request to which the recovery data table was provided as a response.

queryFlag – When present, this element shall have the same value as the Query Flag in the VP1 payload used to form the query request that resulted in this recovery data table.

displayOverride – When present, this element indicates the display override state corresponding to the VP1 payload used to form the query request that resulted in this recovery data table. This element shall have the value 1 if the display override state was true for any part of the Cell conveying that VP1 payload; 0 otherwise.

componentDescription – A data element describing thisComponent in the format defined in Table 5.31 and the parameter descriptions that follow.

querySpread – When present, this element shall express the maximum duration that the receiver is recommended to delay submission of a Dynamic Event HTTP request, in units of 1 millisecond. Specification of a non-zero value expresses a request that upon each instance that the receiver detects a change in the value of the query_flag between successive VP1 payloads in a VP1 Audio Watermark Segment or successive VP1 Message Groups in a VP1 Video Watermark Segment, the receiver is requested to delay the submission of an associated HTTP request for a Dynamic Event by an amount of time with a duration randomly selected with a probability uniformly distributed between 0 milliseconds and the specified value of querySpread, expressed in millisecond increments. The expectation is that the receiver will apply a small enough level of granularity to achieve an even spread of queries across the querySpread duration, such as 1 millisecond.

otherComponent – An element describing another watermarked media component associated with the same service as thisComponent in the format defined in Table 5.31 and the parameter descriptions that follow.

contentID – This field shall identify a content identifier.

contentID.type – A field that is required when contentId element is included. Two values are defined currently:

- "urn:eidr" indicates a content identification per the EIDR registry (http://eidr.org).
- "Designator" for either the "full" or "compact" encoding as defined in SMPTE 2092-1 [16] indicates a content identifier per the Ad-ID registry (http://ad-id.org).

Additional types for user private content identifiers can be used by setting this field to the taggingEntity token as defined in Section 2.1 of IETF RFC 4151 [11], e.g. "atsc.org,2016".

Additional Content ID system types may be defined by ATSC in the future.

`contentID.cid` – A field that is required when `contentId` element is included that provides the content identification. In the case of:

1) "`urn:eidr`": the EIDR Content ID system, this shall be the 34-character canonical form [15] (with hyphens) of the identifier.
2) "Designator" for either the "full" or "compact" encoding as defined in SMPTE 2092-1 [16]: the Ad-ID Content ID system, this shall be the 11-character or 12-character canonical form [16] or the ASCII characters representing the decimal value of the 4-byte unsigned integer Compact Ad-ID Identifier [16].
3) For other domain names, this field is not defined by ATSC. The format of the identifier is determined by the specification of the system.

`contentID.validFrom` – A field that provides information about the UTC time from which the `contentId` is valid on the Recovery Media Timeline.

`contentID.validUntil` – A field that provides information about the UTC time at which the `contentId` will no longer be valid on the Recovery Media Timeline.

`sourceID` – An element describing an attributable distribution source that employs ATSC emission specifications. This element is applicable to circumstances where the watermarked content is included in the redistribution of a service that is broadcast in accordance with ATSC specifications.

`country` – Country code associated with the primary administrative entity under which the value provided in `bsid` is assigned, using the applicable alpha-2 country code format as defined in ISO 3166-1 [12].

`bsid` – This 16-bit unsigned integer field shall identify the Broadcast Stream ID of the original emission signal. In the case that the service is delivered via channel bonding at the physical layer, the indicated BSID shall be the lowest value among the set of BSID values representing the bonded RF emissions.

`majorChannelNo` – The major channel number assigned to the attributable ATSC distribution source. This value is scoped to the BSID.

`minorChannelNo` – The minor channel number assigned to the attributable ATSC distribution source. This value is scoped to the BSID.

`service` – This element describes the service, its signaling formats and broadband locations.

`serviceId` – 16-bit integer that shall uniquely identify this Service within the scope of this Broadcast area.

`sltSvcSeqNum` – This integer number shall indicate the sequence number of the SLT service information with service ID equal to the `serviceId` attribute above. `sltSvcSeqNum` value shall start at 0 for each service and shall be incremented by 1 every time any attribute or child of this `service` element is changed. If no attribute or child element values are changed compared to the previous `service` element with a particular value of `serviceID` then `sltSvcSeqNum` shall not be incremented. The `sltSvcSeqNum` field shall wrap back to 0 after reaching the maximum value.

`slsProtocol` – Specifies the signaling format associated with this service, with permitted values and their meanings defined in Table 5.15.

`slsMajorProtcolVersion` – Major version number for the signaling protocol specified in `slsProtocol`.

`slsMinorProtocolVersion` – Minor version number for the signaling protocol specified in `slsProtocol`.

globalServiceID – A globally unique URI that shall indicate the globally unique Service ID associated with the currently-selected service; for an ATSC 3.0 service this shall have the meaning given for SLT.Service@globalServiceID in A/331 [1] Section 6.3.

svcInetUrl – Base URL to access files for this service via broadband, if available.

urlType – Type of files available with svcInetUrl. See Table 5.9 for values.

urlValue – URL to access Internet signaling files for this service identified by service identifier **serviceId**. Note that some of the signaling files referenced by this attribute, for example, the AEAT, could apply to other services simultaneously.

Table 5.31 Component Description Type

Element or Attribute Name	Use	Data Type	Format
componentDescription			
serverCode	0..1	integer	The Server Code of the component's VP1 watermarks. It is allowed to be present only for the otherComponent element.
componentAnchor	1		The mapping between the first VP1 payload in this VP1 Watermark Segment and its presentation time on the Recovery Media Timeline as defined in Table 5.32.
mediaType	1	string	The media type of this component. Three values are defined: "audio" indicates an audio component, "video" indicates a video component, and "both" indicates that serverCode and intervalCode are present in both audio and video content.
componentID	0..1	string	The identifier of this component.
descriptor	0..1	string	Information associated with the component intended for consumption by an application.
priority	0..1	integer	The relative priority of the component that carries serverCode and intervalCode. Recovery data indicated in components with larger priority values shall take precedence over that of components with lower priority values. When no priority is indicated, the default priority of a component is 0.

componentDescription – A description of a watermarked media component associated with a service.

componentAnchor – Information about the first VP1 payload in the watermarked media component as defined in Table 5.32.

mediaType – A string with value "audio" to indicate that the description applies to an audio component only, "video" to indicate that the description applies to a video component only, or "both" to indicate that the description applies to both an audio and video component.

componentID – A string representing the identifier of this component. The value of this field shall be the same as the **AdpatationSet**@id, **ContentComponent**@id, **Representation**@id in the DASH MPD, or @componentId in User Service Bundle Description Fragment for MMT as specified in Table 7.4 in A/331 [1], corresponding to this component.

descriptor – An arbitrary descriptive string associated with the watermarked media component intended for consumption by an application.

priority – A numeric value indicating the relative priority of the described component. When a receiver finds watermarks in more than one media component in content that it receives, it is expected to perform the recovery process and utilize signaling associated with the component with the highest priority value. When no priority value is indicated for a component, its priority shall be 0.

Table 5.32 Component Anchor

Element or Attribute Name	Use	Data Type	Format
`componentAnchor`	1		
`intervalCodeAnchor`	1	integer	The Interval Code to be used as anchor for presentation time.
`presentationTime`	1	integer	Presentation time in seconds corresponding to `IntervalCodeAnchor`, encoded as the least-significant 32 bits of the count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI).
`presentationTimeMs`	1	integer	milliseconds offset from the time indicated in `presentationTime`
`systemTime`	1	object	Further information related to `presentationTime`, expressed using the `SystemTime` element object specified in A/331 **[1]**.

`componentAnchor` – An element that specifies characteristics of the first VP1 payload in a VP1 Watermark Segment.

`intervalCodeAnchor` – The `intervalCode` (as defined in Section 5.4.1) in the first VP1 payload in a VP1 Watermark Segment.

`presentationTime` – The presentation time on the Recovery Media Timeline of the first frame of the first VP1 Message Group in the VP1 video Watermark Segment, or, for audio components, the presentation time on the Recovery Media Timeline of the first sample of the first symbol in the first Cell of the VP1 Audio Watermark Segment, encoded as a 32-bit count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI).

`presentationTimeMs` –This integer in the range 0 to 999 shall indicate the milliseconds offset from the time indicated in `presentationTime`, such that the formula `presentationTime` + (`presentationTimeMs` / 1000) yields the actual presentation time to the nearest 1 millisecond.

`systemTime` – Further information related to `presentationTime`, based on the `SystemTime` element specified in A/331 [1]. However, the `leap59` and `leap61` name/value pairs of `systemTime` shall indicate the presence of a leap second at the next leap second opportunity which follows `presentationTime` (i.e., in the last minute of the first subsequent UTC day of June 30 or December 31). The `systemTime` object is expected to be used by receivers to perform the computation related to UTC times in the recovered signaling.

5.4.4 Dynamic Event Retrieval via Broadband

Availability of a Dynamic Event is indicated by a change in the query_flag value between successive VP1 payloads within a VP1 Audio Watermark Segment or between subsequent VP1 Message Groups in a VP1 Video Watermark Segment. A change in the query_flag value can also indicate the availability of an updated Recovery File, which can occur due to an unscheduled change to the Content ID List or a change to the Other Component list.

A Dynamic Event request can be made to an Event server by a receiver via issuance of a HTTPS GET request for a Dynamic Event file.

When a receiver is receiving VP1 messages in audio watermarks or video watermarks, the Dynamic Event file URL shall be constructed from a VP1 payload whose query_flag value differs from the preceding payload using the URL template:

```
https://{hostName}/a336/dyn/{subdName}/{serverCode}-
{intervalCode}.dyn[?query]
```

where:

`[string]` items in square brackets [..] indicate an optional `string` and

`{element}` items in curly brackets {..} indicate a named `element`.

The `hostName`, `subdName`, `serverCode` and `intervalCode` shall have the meanings given in Section 5.4.1 and `hostName` shall be determined using the method given in Section 5.4.2. Note that future versions of this standard might define a query string (see RFC 3986 [10]).

When the fingerprint method is used, the Dynamic Event file URL shall be constructed from a response of a fingerprint server using the URL template:

```
{Dynamic Event Base URL}/{intervalCode}.dyn[?query]
```

where:

`[string]` items in square brackets [..] indicate an optional `string` and

`{element}` items in curly brackets {..} indicate a named `element`.

The response to the Dynamic Event request shall contain one or more Dynamic Events. Note that future versions of this standard might define a query string (see RFC 3986 [10]).

In the case when the Event is intended for an Application (i.e., the case when the `schemeIdUri` of the Event is not recognized as the `schemeIdUri` for an Event defined in a standard and directed to the device itself), it should be passed into any Application in the runtime environment that has registered a listener for it, as specified in A/344 [1].

The format of the Dynamic Event shall be as defined in Table 5.14 of this document, and may be delivered directly in the response or may alternatively be packaged together with zero or more other signaling files delivered as defined in Section 5.4.1 of this document. Broadband delivery of Dynamic Event Messages shall employ the media type definition specified in Annex E.2. Dynamic Events need not be signed.

The attributes of an Event that announces updates to signaling files (the `schemeIdUri`, value, etc.) are specified in A/337 [6], with one addition. The abbreviation "RRF" shall be added to the list of signaling tables for which updates can be announced, standing for "Redistribution Recovery File."

5.4.5 AEAT Retrieval via Broadband

The Advanced Emergency Alert Table (AEAT) is delivered as part of Low-Level Signaling (LLS) described in Section 6 of A/331 [1]. Thus, it is not available as part of the Service Layer Signaling (SLS) broadband delivery provided through the signaling server mechanism (A/331 Section 6.7).

A separate URI type is provided to allow the AEAT, if any, associated with the service to be retrieved. In turn, the AEAT could include URLs referencing enhancement content for the event. The complete syntax of the AEAT XML fragment provided over broadband is described in A/331 Section 6.5 [1].

The AEAT URL scheme shall be HTTPS.

Receivers are expected to retrieve the resource identified by an AEAT URL each time a Recovery File containing this type of URI (**`uri_type`** = 0x05) is obtained. This enables a service to use the mechanism for dynamic update of the Recovery File to deliver dynamic updates to the AEAT.

Annex A: Implications of Set-top Box Operations

The purpose of ATSC 3.0 video and audio watermarks (WM) and fingerprints (FP) is to enable receivers to retrieve missing components of ATSC 3.0 services via alternative networks such as broadband to the extent possible. Some of the service components recovered via the WMs or FPs may cause the receiver to modify or replace the video and/or audio components received from the STB.

When a viewer activates a STB function upstream from a receiver, the output from the STB may be modified. For example, bringing up an interactive overlay may modify the video and selecting alternate audio tracks such as alternative language or director's commentary may modify the audio. The viewer expects the STB function to happen, even when WM or FP content recovery systems are activating a service function in the receiver.

Video and audio WMs or FPs allow receivers to identify when a component has been modified by the STB and ensure that viewer expectations are met. For most cases, video and audio are both modified so either audio or video WMs or audio or video FPs can be used. For the full screen UI case, a STB modifies only the video so the video WM/FP is needed. For the partial screen UI case, the STB function may be undetectable using the watermark, depending on the type and amount of video modification. For the mute case, the STB modifies only the audio so the audio WM or FP is needed. Table A.1 shows when the WM or FP has identified when a component has been modified per STB function.

Table A.1 Components Modified and Associated WM/FP Identification Capabilities for STB Functions

STB Function	Video	Video WM/FP	Audio	Audio WM/FP
Channel Change	Modified	Identified	Modified	Identified
Full Screen UI	Modified	Identified	Not Modified	Not Identified
Partial Screen UI	Modified	May Be Identified	Not Modified	Not Identified
Mute	Not Modified	Not Identified	Modified	Identified
Pause	Modified	Identified	Modified	Identified
Rewind	Modified	Identified	Modified	Identified
Fast Forward	Modified	Identified	Modified	Identified
Skip Back	Modified	Identified	Modified	Identified
Skip Ahead	Modified	Identified	Modified	Identified

STB functions need to be detected for service functions that they might conflict with. Different types of service functions need different types of watermarking to identify the relevant STB functions. This informs broadcaster decisions about which watermarks to use and receiver decisions about which service functions to activate. This also influences an interactive application's decisions about which content modifications to perform. Table A.2 shows when a WM or FP is needed for the appropriate service function type.

Table A.2 WM or FP Required to Identify STB Modifications for Receiver Service Functions

Service Function Type	Video WM/FP	Audio WM/FP
Background (e.g. Usage Measurement)	Optional (Required if Audio WM/FP not present)	Optional (Required if Video WM/FP not present)
Visual (e.g. Interactive Overlay)	Required	Optional
Audio (e.g. Dialog Replacement)	Optional	Required
Audiovisual (e.g. Ad Replacement)	Required	Required

Annex B: JSON Recovery File Schema

Normative JSON schema for Recovery File format is as follows:

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "id": "http://atsc.org/version/3.0/wm/recoveryfileformat#",
   "title": "Recovery File format schema",
   "description": "Return messages from server use this schema.",
   "@context": {"RecoveryDataTable": "http://www.atsc.org/contexts/3.0/RecoveryFFV1"},
   "type": "object",
   "properties": {"RecoveryDataTable": {
      "type": "object",
      "properties": {
         "thisComponent": {
            "type": "object",
            "properties": {
               "serverCode": {"type": "integer"},
               "intervalCode": {"type": "integer"},
               "queryFlag": {
                  "type": "integer",
                  "minimum": 0,
                  "maximum": 1
               },
               "displayOverride": {
                  "type": "integer",
                  "minimum": 0,
                  "maximum": 1
               },
               "componentDescription": {
                  "type": "object",
                  "properties": {
                     "componentAnchor": {
                        "type": "object",
                        "properties": {
                           "intervalCodeAnchor": {
                              "type": "integer",
                              "minimum": 0,
                              "maximum": 33554431
                           },
                           "presentationTime": {
                              "type": "integer",
                              "minimum": 0,
                              "maximum": 4294967295
                           },
                           "presentationTimeMs": {
                              "type": "integer",
                              "minimum": 0,
                              "maximum": 999
                           },
```

```
            "systemTime": {
                "type": "object",
                "properties": {
                    "currentUtcOffset": {
                        "type": "integer",
                        "minimum": 0,
                        "maximum": 255
                    },
                    "ptpPrepend": {
                        "type": "integer",
                        "minimum": 0,
                        "maximum": 65535
                    },
                    "leap59": {"type": "boolean"},
                    "leap61": {"type": "boolean"},
                    "utcLocalOffset": {"type": "string"},
                    "dsStatus": {"type": "boolean"},
                    "dsDayOfMonth": {
                        "type": "integer",
                        "minimum": 1,
                        "maximum": 31
                    },
                    "dsHour": {
                        "type": "integer",
                        "minimum": 0,
                        "maximum": 24
                    }
                },
                "required": [
                    "currentUtcOffset",
                    "utcLocalOffset"
                ]
            }
        },
        "required": [
            "intervalCodeAnchor",
            "presentationTime",
            "presentationTimeMs"
        ]
    },
    "mediaType": {
        "type": "string",
        "enum": [
            "audio",
            "video",
            "both"
        ]
    },
    "descriptor": {"type": "string"},
    "priority": {
        "type": "integer",
        "minimum": 0,
        "maximum": 255
    },
    "componentID": {
        "type": "string"
```

```
                }
                },
                "required": [
                    "componentAnchor",
                    "mediaType"
                ]
            }
        },
        "required": ["componentDescription"]
    },
    "querySpread": {"type": "integer"},
    "otherComponent": {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "componentAnchor": {
                    "type": "object",
                    "properties": {
                        "intervalCodeAnchor": {
                            "type": "integer",
                            "minimum": 0,
                            "maximum": 33554431
                        },
                        "presentationTime": {
                            "type": "integer",
                            "minimum": 0,
                            "maximum": 4294967295
                        },
                        "presentationTimeMs": {
                            "type": "integer",
                            "minimum": 0,
                            "maximum": 999
                        },
                        "systemTime": {
                            "type": "object",
                            "properties": {
                                "currentUtcOffset": {
                                    "type": "integer",
                                    "minimum": 0,
                                    "maximum": 255
                                },
                                "ptpPrepend": {
                                    "type": "integer",
                                    "minimum": 0,
                                    "maximum": 65535
                                },
                                "leap59": {"type": "boolean"},
                                "leap61": {"type": "boolean"},
                                "utcLocalOffset": {"type": "string"},
                                "dsStatus": {"type": "boolean"},
                                "dsDayOfMonth": {
                                    "type": "integer",
                                    "minimum": 1,
                                    "maximum": 31
                                },
```

```
                    "dsHour": {
                        "type": "integer",
                        "minimum": 0,
                        "maximum": 24
                    }
                },
                "required": [
                    "currentUtcOffset",
                    "utcLocalOffset"
                ]
            }
        },
        "required": [
            "intervalCodeAnchor",
            "presentationTime",
            "presentationTimeMs"
        ]
    },
    "mediaType": {
        "type": "string",
        "enum": [
            "audio",
            "video",
            "both"
        ]
    },
    "descriptor": {"type": "string"},
    "priority": {
        "type": "integer",
        "minimum": 0,
        "maximum": 255
    },
    "componentID": {
    "type": "string"
    },
    "serverCode": {
        "type": "integer",
        "minimum": 0,
        "maximum": 2147483647
    }
},
"required": [
    "componentAnchor",
    "mediaType"
]
},
"minItems": 0
},
"contentID": {
"type": "array",
"items": {
    "type": "object",
    "oneOf": [
        {
            "properties": {
                "type": {
```

```
                "type": "string",
                "enum": ["urn:eidr"]
            },
            "cid": {
                "type": "string",
                "pattern": "^10\\.5240\\/([0-9a-fA-F]{4}-){5}[0-9A-Z]$",
                "minLength": 34,
                "maxLength": 34
            },
            "validFrom": {
                "type": "string",
                "format": "date-time"
            },
            "validUntil": {
                "type": "string",
                "format": "date-time"
            }
        },
        "required": [
            "type",
            "cid",
            "validFrom"
        ]
    },
    {
        "properties": {
            "type": {
                "type": "string",
                "enum": ["urn:smpte:ul:060E2B34.01040101.01200900.00000000"]
            },
            "cid": {
                "type": "string",
                "pattern": "^[1-9a-zA-Z]{1}[0-9a-zA-Z]{10}(H|D)?$",
                "maxLength": 12
            },
            "validFrom": {
                "type": "string",
                "format": "date-time"
            },
            "validUntil": {
                "type": "string",
                "format": "date-time"
            }
        },
        "required": [
            "type",
            "cid",
            "validFrom"
        ]
    },
    {
        "properties": {
            "type": {
                "type": "string",
                "enum": ["urn:smpte:ul:060E2B34.01040101.01012009.00000000"]
            },
```

```
                "cid": {
                    "type": "string",
                    "pattern": "^[0-9]{1,10}$",
                    "maxLength": 10
                },
                "validFrom": {
                    "type": "string",
                    "format": "date-time"
                },
                "validUntil": {
                    "type": "string",
                    "format": "date-time"
                }
            },
            "required": [
                "type",
                "cid",
                "validFrom"
            ]
        },
        {
            "properties": {
                "type": {
                    "type": "string",
                    "not": {"enum": [
                        "urn:eidr",
                        "urn:smpte:ul:060E2B34.01040101.01200900.00000000",
                        "urn:smpte:ul:060E2B34.01040101.01012009.00000000"
                    ]}
                },
                "cid": {"type": "string"},
                "validFrom": {
                    "type": "string",
                    "format": "date-time"
                },
                "validUntil": {
                    "type": "string",
                    "format": "date-time"
                }
            },
            "required": [
                "type",
                "cid",
                "validFrom"
            ]
        }
    ]
},
"minItems": 0
},
"sourceID": {
    "type": "object",
    "properties": {
        "country": {
            "type": "string",
            "pattern": "^[a-zA-Z]{2}$"
```

```
            },
            "bsid": {
                "type": "integer",
                "minimum": 0,
                "maximum": 65535
            },
            "majorChannelNo": {
                "type": "integer",
                "minimum": 1,
                "maximum": 999
            },
            "minorChannelNo": {
                "type": "integer",
                "minimum": 1,
                "maximum": 999
            }
        },
        "required": [
            "country",
            "bsid",
            "majorChannelNo",
            "minorChannelNo"
        ]
    },
    "service": {
        "type": "object",
        "properties": {
            "serviceId": {
                "type": "integer",
                "minimum": 0,
                "maximum": 65535
            },
            "sltSvcSeqNum": {
                "type": "integer",
                "minimum": 0,
                "maximum": 255
            },
            "slsProtocol": {
                "type": "integer",
                "minimum": 0,
                "maximum": 255
            },
            "slsMajorProtocolVersion": {
                "type": "integer",
                "minimum": 0,
                "maximum": 255
            },
            "slsMinorProtocolVersion": {
                "type": "integer",
                "minimum": 0,
                "maximum": 255
            },
            "globalServiceID": {
                "type": "string",
                "format": "uri"
            },
```

```
                "svcInetUrl": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "urlType": {
                                "type": "integer",
                                "minimum": 0,
                                "maximum": 255
                            },
                            "urlValue": {
                                "type": "string",
                                "format": "uri"
                            }
                        },
                        "required": [
                            "urlType",
                            "urlValue"
                        ]
                    }
                }
            },
            "required": [
                "serviceId",
                "sltSvcSeqNum"
            ]
        }
    },
    "required": [
        "thisComponent",
        "service"
    ],
    "additionalProperties": true
}},
"required": ["RecoveryDataTable"]
}
```

Annex C: Acquisition of Signaling Using Video Watermarks

The following is a summary of the signaling acquisition process using video watermarks excluding the case where VP1 Payloads are used in the video watermarks, which is covered in Annex D:

- The device acquires video watermarks with a Content ID Message payload and a signaling URI Message payload.
- The device uses the signaling URI Message from the signaling URI Message payload to obtain the signaling files in the form of a multi-part MIME message, exactly as it would use the signaling URL in the audio watermark Recovery File.
- The device uses the signaling files to access and present the supplementary content using timing information from a Presentation Time Message payload to synchronize the content with the audio and video coming from the cable.
- Each signaling file has a `validFrom`, a `validUntil`, and a `nextURL` attribute associated with it. The `validFrom` and `validUntil` fields define the interval of validity of the signaling file on the Recovery Media Timeline, and the `nextURL` is the URL of the next scheduled version of that signaling file. Thus, the device can get scheduled updates to the signaling files as needed.
- From time to time Dynamic Events may appear in a Dynamic Event Message payload of a video watermark.
- Such an event can be intended for an application running in the run-time environment, or it can be signaling the availability of an unscheduled update to a signaling file. In the former case, the device can just make the event available to any application that has registered a callback routine for it, in the usual way. In the latter case the event will contain the URL of the updated signaling file (or files), and the device can retrieve it (or them). The file or files will be in the same format as files retrieved by a signaling URL, with `validFrom` and `validUntil` and `nextURL` attributes attached to them.
- If at any time the video watermarks disappear, the device is expected to terminate presentation of any supplementary content, interpreting the disappearance as evidence of a channel change to a new service that does not support any supplementary content.
    - Note that if audio watermarks are also present and have not simultaneously disappeared, this may be an indication of the presence of graphics generated by the set-top box, such as displaying the MVPD program guide, rather than a channel change. Please see Annex A for more information.

If all Content ID Messages and VP1 watermarks disappear, or a Content ID Message watermark or VP1 watermark appears indicating a new combination of BSID and channel number, or a discontinuity is detected in the Recovery Media Timeline, then the receiver is expected to suspend presentation of any supplementary content, and start the signaling acquisition over again.

Annex D: Acquisition of Signaling Using VP1 Payloads/Messages

The following is a summary of the watermark-based signaling acquisition process via broadband, starting from a VP1 audio watermark payload or VP1 video watermark message:

- The device recovers the VP1 payload from the audio or video watermark.
- The device makes an HTTPS request to a Recovery File Server, using the domain type, server field and the interval field of the VP1 payload to form the URL for the request (defined in Section 5.4.1).
- The device receives in response a Recovery File (format defined in Section 5.4.3) that includes a mapping of the Recovery Media Timeline to the position in content of the start of the VP1 payload, information about the service being viewed, and a URL that can be used to retrieve the set of signaling files needed to access and present the supplementary content. These signaling files will be current as of the position in the content of the VP1 payload.
- The device uses the URLs in the Recovery File to retrieve the appropriate set of signaling files, e.g., MPD, HELD, AEAT, AEI, MPT, etc.
- The device uses the signaling files to access and present the supplementary content, using the timing information in the Recovery File to synchronize the content with the audio and video coming from the cable.
- The Query Flag in the VP1 payload is used to signal the availability of an event. When a change of value of the Query Flag is detected, the device can request the event, using information from the VP1 payload (defined in Section 5.4.4).
- Such an event can be intended for an application running in the run-time environment, or it can be signaling the availability of an unscheduled update to a signaling file. In the former case, the device can make the event available to any application that has registered a callback routine for it, in the usual way. In the latter case the event contains the URL of the updated signaling file (or files), and the device can retrieve it (or them).
- The Recovery File, Dynamic Event, and signaling files are delivered as a multi-part MIME message encapsulated in an MBMS “metadata envelope” (defined in Section 6.7 of A/331 [1]) that includes a “valid from” and a “valid until” and a “next URL” attribute associated with each signaling file. The “valid from” and “valid until” attributes define the interval of validity of the signaling file, and the “next URL” attribute is the URL of the next scheduled version of that signaling file. Thus, the device can get scheduled updates to the signaling files as needed.
- If at any time the audio watermarks disappear, the device will terminate presentation of any supplementary content, interpreting the disappearance as evidence of a channel change to a new service that does not support any supplementary content.

  - Note that if video watermarks are also present and have not simultaneously disappeared, this may be an indication of muting at the set-top box, rather than a channel change. Please see Annex A for more information.
- If at any time the server field in the audio watermarks changes, or the interval field has a discontinuity, the device will terminate presentation of any supplementary content, request a new Recovery File and restart the recovery process.

In the case of a ROUTE/DASH-based service, the Recovery Media Timeline will correspond to the media presentation timeline of the current MPD for the service. In the case of an MMT-based service, the Recovery Media Timeline will correspond to the MMT presentation timeline.

Annex E: Media Type Registrations

This Annex documents new media types registered by IANA at https://www.iana.org/assignments/media-types/media-types.xhtml#application.

> *Notice to editors*: any changes to this Annex are subject to review by IETF and IANA as described in IETF BCP 13 [20].

E.1 RDT

Type name:

application

Subtype name:

atsc-rdt+json

Required parameters:

N/A

Optional parameters:

N/A

Encoding considerations:

binary

Constrained to UTF-8.

See RFC 8259 [17], Section 8.1.

Security considerations:

This media type inherits the issues common to all JSON media types - see RFC 8259 [17], Section 12.

This media format is used to describe broadcast and broadband services. This format is highly susceptible to manipulation or spoofing for attacks desiring to mislead a receiver about a URL for signaling, reporting, or other services. Both integrity protection and source authentication are recommended to prevent misleading of processors. This type does not employ executable content, but since it is explicitly extensible then executable content could appear in an extension.

This media type does not provide any confidentiality protection and instead relies on the transport protocol that carries it to provide such security, if needed.

This media type also includes URL references which if present must be dereferenced in order to properly interpret the content. This creates additional general security considerations. Decoders should make every effort to validate the origin against other transport signaling, if present.

Interoperability considerations:

ATSC A/336 specifies the format of conforming messages and the interpretation thereof.

Published specification:

This media type registration is an integral part of ATSC A/336, "Content Recovery in Redistribution Scenarios", Annex E. The payload is defined in Section 5.4.3 of that document. This specification, which contains the JSON schema for the content in Annex B, is available at www.atsc.org/standards.

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

N/A

Person & email address to contact for further information:

Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)

Intended usage:

COMMON

Restrictions on usage:

N/A

Author:

ATSC.

Change controller:

ATSC.

E.2 DYNAMIC EVENT MESSAGE

Type name:

application

Subtype name:

atsc-dynamic-event-message

Required parameters:

N/A

Optional parameters:

N/A

Encoding considerations:

binary

This media type may require non-transparent transfer encoding (such as base64 or Quoted-Printable) on transports not capable of handling binary.

Security considerations:

This media type identifies a dynamic event message, which is used to deliver a stream of dynamic events, such as a list of updated signaling tables. Events themselves are notifications saying that some action needs to be taken.

Since actions can cause receivers to behave in fairly arbitrary ways, both integrity protection and source authentication, as referenced in A/336 Section 5.4, are required to prevent misleading of processors.

This media type does not provide any confidentiality protection and instead relies on the transport protocol that carries it to provide such security. These protocols are referenced in A/336 Section 5.4.

Dynamic event messages are retrieved from an event server by dereferencing a URL that is constructed by the Receiver using the process defined in A/344 Section 5.4 and is in a domain specified and administered under authority of ATSC. The only domain currently allowed is vp1.tv, as defined in A/336 Table 5.20. This mitigates additional general security considerations, as signaling resulting in an atsc-dynamic-event-message can only be resolved to a server having a valid, active registration with the VP1 Registrar. In addition to this, processors should make every effort to validate the origin against other transport signaling, if present.

Interoperability considerations:

ATSC A/336 specifies the format of conforming messages and the interpretation thereof.

Published specification:

This media type registration is an integral part of ATSC A/336, "Content Recovery in Redistribution Scenarios", Annex E. The payload is defined in Section 5.1.8 of that document. This specification is available at https://www.atsc.org/atsc-documents/type/3-0-standards.

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

N/A

Person & email address to contact for further information:

Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)

Intended usage:

COMMON

Restrictions on usage:

N/A

Author:

ATSC.

Change controller:

ATSC.

— End of Document —